(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 April 2001 (12.04.2001)

PCT

(10) International Publication Number
WO 01/26378 A1

(51) International Patent Classification[7]: H04N 7/24

(21) International Application Number: PCT/US00/27634

(22) International Filing Date: 5 October 2000 (05.10.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/157,998 6 October 1999 (06.10.1999) US

(71) Applicant: STREAMING21, INC. [US/US]; Suite 1, 170 Knowles Drive, Los Gatos, CA 95032 (US).

(72) Inventors: LEE, Yen-Jen; 3643 Madison Common, Fremont, CA 94538 (US). SHIM, Sang, Yup; 1157 Pheasant Hill Drive, San Jose, CA 95120 (US).

(74) Agents: LEBLANC, Stephen, J. et al.; Majestic, Parsons, Siebert & Hsue P.C., Suite 1100, Four Embarcadero Center, San Francisco, CA 94111-4106 (US).

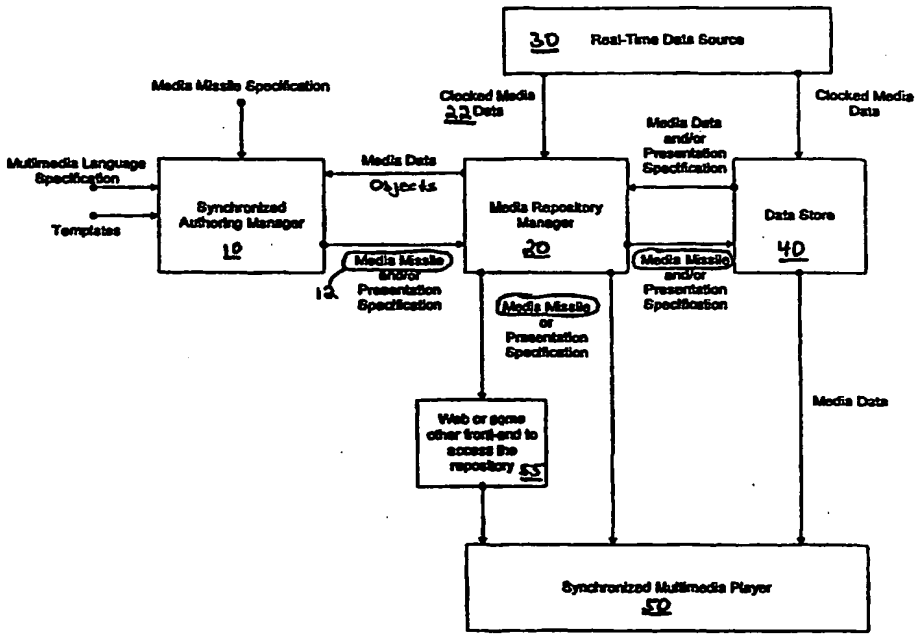
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*
— *Before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHOD AND APPARATUS FOR MANAGING STREAMING DATA



(57) Abstract: A streaming media structure methods and systems for creating, managing, and delivering streaming media is disclosed. The streaming media structure is provided to create, deliver, reassemble, and render synchronized multimedia content with needed machine-aware components through streaming services.

WO 01/26378 A1

METHOD AND APPARATUS FOR MANAGING STREAMING DATA

CROSS REFERENCE TO RELATED APPLICATIONS

This application claims priority from provisional patent application 60/157,998 entitled STREAMING MACHINE-AWARE BINARY AND MULTIMEDIA CONTENT FOR MULTIMEDIA SYNCHRONIZATION filed 10/06/99.

The above referenced application(s) are incorporated herein by reference for all purposes. These prior applications, in some parts, may indicate earlier efforts at describing the invention or describing specific embodiments and examples. The present invention is therefore best understood as described herein.

FIELD OF THE INVENTION

The invention generally relates to methods and/or devices related to data transmission and presentation using a computing device or information appliance. More particularly, the invention in various aspects relates to multimedia data methods and applications and to a versatile file structure suitable for providing multimedia data and streaming data over a communication data network.

BACKGROUND OF THE INVENTION

Previous works related to aspects of the present discussion include below listed reference by Blakowski et al. and by the WWW Consortium. The Blakowski reference provides a survey of research regarding media synchronization. The reference discusses a four-layer synchronization reference model. From the lower layer to the upper layer are Media Layer, Stream Layer, Object Layer, and Specification Layer. A variety of techniques have been published and implemented in the specification area. SMIL is one of the latest efforts in the industry to provide a presentation specification using descriptive tagged language, which follows the syntax of XML.

Existing and popular media container formats for computers include the QuickTime from Apple Computer (http://www.apple.com), ASF/AVI from Microsoft (http://www.microsoft.com), RealMedia from RealNetworks (http://www.real.com), and Video Capsule from Digital Lava (http://www.digitallavacom). Some of these such as

RealMedia and Video Capsule do not have published technical specifications. Some existing formats, such as *.exe* video files, provide a means for distributing media clips that are executable and viewable in a standard operating system without the need for installing additional software. In such a file, necessary software components are included in a single file with a name like *movieclip.exe*. Generally, such a file is not capable of playing streaming data or of varying the components downloaded.

Downloadable machine code technology (e.g. applets) is also known. Sun pioneered aspects of downloadable executable components by creating Java, which has become used to make software components downloadable through web browsers to run on a local virtual machine. Microsoft's COM technology is a later developed technology ready to be downloaded from Internet and run in a document container, such as a web browser.

The Internet is a rapidly growing communication network of interconnected computers around the world. Together, these millions of connected computers form a vast repository of hypermedia information that is readily accessible by users through any of the connected computers from anywhere and anytime. As there are an increasing number of users who are connected to the Internet and surf for various information, there is tremendous demand for more content to be available and methods to deliver content on the Internet. Currently, the most commonly available media information that is deliverable on the Internet may include text information, images and graphics, videos and audio clips.

One known format for this data is streaming data, often used for video and/or audio data. For example, streaming video is a sequence of encoded images sent over a network or other communications media and presented at a user device, with presentation of part of the data overlapping some of the receiving of the data. With streaming data, a user does not have to wait to completely download a large file before accessing that file or parts thereof (e.g. seeing video or hearing sound). As will be understood in the art, streaming data, such as video data, can be "live" data, such as streaming encoding of a radio or television broadcast, or can be previously stored data, such as pre-recorded audio or video data. The designation "real-time" is sometimes used in the art to indicate streaming data generally, included pre-stored media components. This term is used to indicate that delivery of the data is time-critical in that once playback begins, the

streaming data must be delivered quickly enough to keep up with "real-time" presentation of the data to the user. Such data is sometimes also referred to in the art as "clocked data" to indicate that the data generally must be delivered at a nearly constant rate. Too-fast delivery can swamp the receiving end's buffer capacity. Too slow delivery will cause unacceptable halts or pauses in playback.

To access streaming data, a user generally uses an access module. This module can be either software or hardware, with logic functions able to present the data in a form desired by a user. As examples, an access module could be a software player (such as RealMedia™) running on a compatible information platform. An access device may also be more dedicated hardware, such as a cell-phone capable of accessing certain types of streaming data, a PDA, a television with the necessary logic, etc. One difficulty with standard access methods is that with the increased improvement in video/audio compression technologies, access module logic may have to be updated frequently when new data formats are used. Further, for media presentations including different types of data, an access device (such as a general purpose computer or PDA) has sometimes to be loaded with several different access modules to access different media.

SMIL and SMIL Players and Authoring Tools

To facilitate the creation and distribution of multimedia content on the web, in June of 1998, the World Wide Web Consortium (W3C) announced the first specification of the Synchronized Multimedia Integrated Language (SMIL). This language provides some mechanisms for structuring various elements of multimedia content, including audio, text, image, and motion video and provides a way for web authors to schedule the playing of time-based media within HTML documents.

A SMIL document according to the specification is actually an ASCII text file, which authors can edit with almost any text editor. SMIL documents generally resemble HTML files and references included components from external URL locations. SMIL uses XML to provide the mechanism to define SMIL markup tags for associating timing and positioning relationships between multimedia objects, so that audio, videos, images, and text can be synchronized.

SMIL enables web authors to position (location, size, z-index ordering) visual media objects and assign temporal attributes (begin time, duration or end time) among one and another. SMIL provides capabilities to create interactive (via hyperlinks)

multimedia presentations similar to those on computer-based interactive CDs. SMIL media objects (addressable by a URL) can either reside locally or be distributed over the web. SMIL media objects can be updated independently and remotely.

A SMIL Player is used to schedule multimedia presentation in a SMIL document, retrieving media objects on the web using URLs (Universal Resource Locator) described in the document, parsing the timing relationship among downloaded media, and consequently displaying the presentation based on the correct timeline. The Java SMIL Player is a Java applet-based player that follows a platform-neutral programming paradigm and makes use of the XML Parser and Java Media Framework (JMF). It can run within any popular browser without the need to download any plug-ins or ActiveX controls.

A number of SMIL authoring tools have been proposed, which can generally be grouped into two categories: general purpose tools designed to create synchronized multimedia content for any SMIL-compliant players; and special purpose tools designed to create synchronized multimedia content for particular players. While these authoring tools are powerful, they still require proficiency in SMIL programming, which limits the range of users. Currently most SMIL authoring tools in the market have several properties in common: they include a text-based editor; they include a WYSIWYG (What You See Is What You Get) layout editor; and they do not include a media timeline design wizard. T.A.G. SMIL Editor 1.0, for example, is a general purpose SMIL authoring tool, which is designed to create synchronized multimedia content for any SMIL-compliant players. SuperTool is a specific purpose SMIL authoring tool for Real G2 Player only. SuperTool has a point and click WYSIWYG layout interface. Developers can add available media types, arrange media's layout and sequence how they are played in SMIL composition.

Prior Publications

The following publications may be related to the invention or provide background information. Listing of these references here should not be taken to indicate that any formal search has been completed or that any of these references constitute prior art.

Blakowski, G, and Steinmetz, R. A Media Synchronization Survey: Reference Model, Specification, and Case Studies, IEEE Journal on Selected Areas in Communications, Vol. 14, No. 1, pp. 5-35, Jan. 1996.

Synchronized Multimedia Integration Language (SMIL) 1.0 Specification, WWW Consortium, Jun. 1998, available at www.w3.org/TR/REC-smil.

SUMMARY

The approach of the present invention is different from aforementioned works in part because a Media Missile is open to and can carry different media specification's requirements. In one embodiment, not only actions and data embedded in the media objects are part of the Media Missile, the Media Missile can also carry executable binary, signature, and auxiliary information (which, in specific embodiments, may be pointed to by URLs). Unlike the above mentioned prior streaming techniques, which do not describe carrying various active media content, an executable binary discussed below can be carried in a Media Missile to make it ready for a runtime loader on a presentation device to run the media missile executable without need for other preinstalled software. Thus, in specific embodiments, Media Missile makes uses of operating system's launch sequence to run an included executable component and is self-contained for both of the software components and media data.

Media Missile, according to specific embodiments of the present invention, can benefit from technological advances such as Java or COM, but does not rely on them.

In specific embodiments, the invention involves a method and/or system for providing streaming content that can be activated from a single resource locator and in response to an activation, begin transferring a streaming presentation in a multimedia container, wherein said container has the capability of including one or more executable code components, one or more streaming content components, and one or more sets of presentation data. A Media Missile launcher in further embodiments may receive from a user device data indicating available hardware and software components at said user device and this data may be used by a scheduler/parser at the user device to request or not request different components from the media missile repository. Therefore, a media missile can be transmitted with some executable code or no executable code, depending on which components are present at the user site.

In a further embodiment, presenting a media missile according to the invention is self-directed and requires minimal active management by or presence of preinstalled logic components at a user device.

In further specific embodiments, the invention involves a method and/or system allowing a creator (or author) to specify a streaming presentation including receiving indications from an author specifying one or more items of streaming content and specifying presentation parameters. The content and parameters may be located at a server remote from the author.

The invention and various specific aspects and embodiments will be better understood with reference to the following drawings and detailed descriptions. In different figures, similarly numbered items are intended to represent similar functions within the scope of the teachings provided herein. In some of the drawings and detailed descriptions below, the present invention is described in terms of the important independent embodiment of delivering visual and/or audio content. This should not be taken to limit the invention, which, using the teachings provided herein, can be applied to other streaming data content. For purposes of clarity, this discussion refers to devices, methods, and concepts in terms of specific examples. However, the invention and aspects thereof may have applications to a variety of types of devices and systems. It is therefore intended that the invention not be limited except as provided by the attached claims and equivalents.

Furthermore, it is well known in the art that logic or digital systems and/or methods can include a variety of different components and different functions in a modular fashion. The following will be apparent to those of skill in the art from the teachings provided herein. Different embodiments of the present invention can include different combinations of elements and/or functions. Different embodiments of the present invention can include actions or steps performed in a different order than described in any specific example herein. Different embodiments of the present invention can include groupings of parts or components into larger parts or components different than described in any specific example herein. For purposes of clarity, the invention is described in terms of systems that include many different innovative components and innovative combinations of innovative components and known components. No inference should be taken to limit the invention to combinations containing all of the innovative components listed in any illustrative embodiment in this specification. The functional aspects of the invention, as will be understood from the teachings herein, may be implemented or accomplished using any appropriate implementation environment or

programming language, such as C++, Cobol, Pascal, Java, Java-script, etc. All publications, patents, and patent applications cited herein are hereby incorporated by reference in their entirety for all purposes.

The invention therefore in specific aspects provides a streaming video/audio signal that can be played on various types of video-capable terminal devices operating under any types of operating systems regardless what type of players are preinstalled in the terminal devices

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram showing an overall architecture of a system showing both authoring and playback according to various embodiments of the invention including multiple novel elements.

FIG. 2 illustrates an exemplary configuration of a network in which the present invention may be practiced.

FIG. 3 illustrates an example media missile container format according to specific embodiments of the invention.

FIG. 4 is a flow chart illustrating functional aspects of a multimedia player according to a specific embodiment of the invention.

FIG. 5 is a flow chart illustrating an example method for accessing a media missile according to a specific embodiment of the invention.

FIG. 6 illustrates the operation of different timers initiated by a scheduler according to specific embodiments of the invention.

FIG. 7 shows a standard browser display of example source files that may be packaged into a media missile in an example in accordance with specific embodiments of the present invention.

FIG. 8 shows an example interface for packaging source data into a media missile according to specific embodiments of the present invention.

FIG. 9 illustrates an example general metadata only media missile format according to specific embodiments of the invention.

FIG. 10 illustrates an example metadata only media missile data structure according to specific embodiments of the invention.

FIG. 11 illustrates an example media missile data structure according to specific embodiments of the invention.

FIG. 12 illustrates an example binary encoded metadata only media missile data according to specific embodiments of the invention.

FIG. 13 illustrates an example presentation of a media missile in a player according to specific embodiments of the present invention.

FIG. 14 illustrates an example general method for guiding a user in creating a media missile or presentation data according to specific embodiments of the invention.

FIG. 15 illustrates an example of a Synchronized Authoring Manager (SAM), also referred to as a Synchronized Authoring Tool (SAT), according to specific embodiments of the invention.

FIG. 16 illustrates an example of a Preview Manager GUI according to specific embodiments of the present invention.

FIG. 17A-C illustrate examples of choosing various presentation templates according to specific embodiments of the present invention.

FIG. 18A-B illustrate adding and previewing media according to a Video Lecture Template according to specific embodiments of the present invention.

FIG. 19A-B illustrate adding and previewing media according to a Headline News Template according to specific embodiments of the present invention.

FIG. 20A-B illustrate examples of creating presentation specification metadata according to specific embodiments of the present invention.

FIG. 21 illustrates an example of a Document Generation GUI according to specific embodiments of the present invention.

FIG. 22 is a block diagram showing a representative example logic device in which aspects of the present invention may be embodied.

DESCRIPTION OF SPECIFIC EMBODIMENTS I

In specific embodiments, the present invention involves a streaming media structure suitable for providing multimedia streaming over a communication data network including a cable network, a local area network, a network of other private networks and the Internet. The detailed description of the present invention include numerous specific details that are set forth in order to provide a thorough understanding of the present

invention. However, it will become obvious to those skilled in the art that the present invention may be practiced without these specific details. In other instances, well known methods, procedures, components, and circuitry have not been described in detail to avoid unnecessarily obscuring aspects of the present invention.

The present invention is presented largely in terms of procedures, steps, logic blocks, processing, and other symbolic representations that resemble data processing devices. These process descriptions and representations are the means used by those experienced or skilled in the art to most effectively convey the substance of their work to others skilled in the art. The method along with the system to be described in detail below is a self-consistent sequence of processes or steps leading to a desired result. These steps or processes are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities may take the form of electrical signals capable of being stored, transferred, combined, compared, displayed and otherwise manipulated in a computer system or electronic computing devices. It proves convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, operations, messages, terms, numbers, or the like. It should be borne in mind that all of these similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

According to one aspect of the present invention, a streaming media structure is described below. The streaming media structure is provided to create, deliver, reassemble, and render synchronized multimedia content with its lightweight machine-aware components through streaming services. Each stream represents a user access to a video/audio title provided in a host from a terminal device.

FIG. 2 illustrates an exemplary computer communication configuration in which the present invention may be practiced. Central server 102 together with a video/audio database 104 is a video/audio source comprising video/audio files that can be accessed on demand. As used herein, video/audio files or titles are referred to any video footage, video films and/or video/audio clips that can be compressed or packed in any format. One of the exemplary formats is Moving Picture Experts Group (MPEG). To play MPEG video files, one needs need an MPEG viewer or a client software executing in a personal computer with sufficient processor speed, internal memory. Another one of the popular formats for audio is MPEG-1 Audio Layer-3 (MP3), a standard technology and format for

compressing a sound sequence into a very small file (about one-twelfth the size of the original file) while preserving the original level of sound quality when it is played through a client program (player) downloadable from many web sites. It should be noted, however, that the exact formats of the video files do not affect the operations of the present invention. As will be noted and appreciated, the present invention applies to any formats of the video or audio files as well as other multimedia data that may include, but not be limited to, binary data and files, hypertext files and scripts.

Preferably, data network 106 is a data network backbone, namely a larger transmission line. At the local level, a backbone is a line or set of lines that local area networks connect to for a wide area network connection or within a local area network to span distances efficiently (for example, between buildings). On the Internet or other wide area network, a backbone is a set of paths that local or regional networks connect to for long-distance interconnection. Coupled to data network A 106, there are two representative proxy servers 108 and 110 that each service representative terminal devices 116-119 via data network 112 and 114 respectively. It should be noted server 102 sometimes named "central" or "central video" server does not necessary mean that server 102 is only served as a repository of all the video titles. In some case, server 102 may service terminal devices directly and may retrieve other video/audio files from other servers such as proxy servers 108 and 110.

Data network 112 and 114 are typically the Internet, a local area network or phone/cable network through which terminal devices can receive video files. The terminal devices may include, but not be limited to, multimedia computers (e.g. 116 and 119), networked television sets or other video/audio players (e.g. 117 and 118). Typically the terminal devices are equipped with applications or capabilities to execute and display received video files. For example, one of the popular applications is an MPEG player provided in WINDOWS 98 from Microsoft. When an MPEG video file is received in streaming from one of the proxy servers, by executing the MPEG player in a multimedia computer, the video file can be displayed on a display screen of the computer.

According to one aspect of the present invention, the streaming media structure is provided to create, deliver, reassemble, and render synchronized multimedia content with its lightweight machine-aware components through streaming services.

Cases 1 and 2 below present detailed procedures of how to construct the streaming media structure relying on a complied and linked version implementing the present invention and executing preferably in a client device. As such a video/audio title can be submitted to a host device in a format compliant to the streaming media structure. Cases 3 and 4 below present detailed procedures of how to playback video/audio titles packed in a streaming media data structure according to embodiments of the present invention.

It can be appreciated by those skilled in the art that one of the important features in the present invention is a new versatile streaming data file format or structure that can be accessed from any types of terminal devices operating under any types of operation systems. Another important feature is that the present invention, in specific embodiments, allows the elimination of preinstalled players. A streaming media structure according to specific aspects of the invention can include a built-in player that is downloaded on the fly and then operates in a local machine to access the streaming data and to find and install any necessary additional executable components. As will be understood to those of skill in the art from the teachings herein, elimination of preinstalled players for various types of streaming content allows streaming content to be more easily accessed on devices with limited code storage, such as cell-phones or other hand-held or versatile playback devices.

The processes, sequences or steps and features discussed herein are related to each other and each are believed independently novel in the art. The disclosed processes and sequences may be performed alone or in any combination to provide a novel and nonobvious file structure system suitable for media delivery system. It should be understood that the processes and sequences in combination yield an equally independently novel combination as well, even if combined in their broadest sense.

The streaming media structure as described herein, in accordance with one aspect of the present invention is robust, operationally efficient and cost-effective. The mechanism provides the best use of servers providing streaming services and permits seamless delivery of streaming video with highest quality of services possible. In addition, the present invention may be used in connection with presentations of any type, including sales presentations and product/service promotion, which provides the video service providers additional revenue resources.

In order to facilitate description, the following discussion will describe the present invention in terms of streaming media presentations. It will be understood to those of skill in the art, however, that the invention may be used in other streaming data situations. The invention should therefore not be taken as limited except as provided in the attached claims.

In various embodiments, systems or methods according to various aspects of the present invention, can create, deliver, reassemble, and render synchronized streaming multimedia content with lightweight machine-aware components through streaming services. Each stream represents a user access from a client on one computer to a server on a different computer. As will be understood from the teachings herein, a system or method according to the present invention can handle various types of streaming data including real-time multimedia content as part of a presentation or multimedia missile through streaming services.

In particular embodiments, an author (a user or a program) creates synchronized presentation using a wizard-like Synchronized Authoring Manager (SAM). In further embodiments, an SAM can include lightweight software components and/or indications of such components with the media data content into a Media Missile container that is streamable from a single location.

The present invention, in some embodiments, thus allows that the streaming of active and passive content can be done all at once and can be self-contained and guidance-free. The active content is the various machine-aware runtime components. The passive content is the media data to be rendered at runtime. All of the presentation content (both active and passive) can be stored in a container-like file format or object-relational database. A special header can be attached to individual media objects to specify metadata information such as signature, data length (or offset), and linkage. A Presentation Specification can be generated in descriptive tagged language such as Synchronized Multimedia Integration Language (SMIL) from World-Wide Web Consortium (W3C).

A container-like file format of a Media Missile has advantages for content providers in the convenience and portability of the content. With the flexible file specification of the Media Missile, a content provider of a synchronized Multimedia presentation can create the presentation and then store, manage, distribute, or serve the

presentation as easily as they would a single file. Thus, a synchronized multimedia presentation, using a media missile, can be as easy to deliver or manage as a standard GIF or JPEG banner ad. Conventional SMIL-based or HTML-based presentations cannot merge multiple objects into a single container and can require managing and downloading dozens of individual files to render properly.

A media player implementation, such as a an SMIL player, that is compliant to this design can render and playback media contents in real-time in accordance with a presentation specification. In particular, individual media content objects, specified in the presentation, corresponds to object information in the metadata and media data in the media missile.

In a further embodiment, a one-way one-to-one hashing function or other method can be used to generate a signature for the media. The signature can be both a watermark of the media, and also be a search key (to be used in place of a name) if the media is stored in a large distributed media repository. In a further important embodiment, a media missile format enables synchronized presentation on mobile platform with limited pre-installed software capability or memory footprint.

FIG. 1 is a block diagram showing an overall architecture of a system showing both authoring and playback according to various embodiments of the invention including multiple novel elements. Various aspects and details of the designs are elaborated in later sections. The overall architecture shown in FIG. 1 includes the following functional and/or structural elements.

Media Missile (MM) 12 is a container specification that allows inclusion of various items of machine-aware executable binary (or script), media data, and associated information such as presentation specification or metadata. (MM) can be a container of container(s) in a nested structure. The machine-aware executable binary (or script) is can also be referred to as active media content. The active media content can be implemented in machine-neutral programming paradigm (such as Java) or machine-dependent code (such as .exe files). Media Missile with active media content is self-contained and guidance-free for synchronized real-time multimedia authoring and/or presentation over network environment. Once the media missile is indicated and the active media content is retrieved and loaded to a machine, it is ready to run by itself.

Synchronized Authoring Manager (SAM) 10 takes and understands various language specifications, formats, and templates in synchronized multimedia, including Media Missile. Language specification and templates such as SMIL-compliant ones are used as input to SAM's authoring wizard. In specific embodiments, an SAM wizard guides a user to compose a synchronized presentation and retrieve media data from Media Repository Manager 20. In one option, an end result of an SAM is a packaged Media Missile. As an option, an SAM 10 can also output a separate presentation specification (such as an SMIL presentation specification) that is compliant to international or industry standards.

As shown in FIG. 1, SAM 10 can receive as inputs templates, media missile specification, multimedia language specification, and media data, which may be received from a media repository. Templates, as understood in the art, include data describing a general format, layout, or structure for a presentation. SAM 10 creates a media missile and/or presentation specification, which is stored in repository 20.

Media Repository Manager (MRM) 20 handles storage and retrieval of real-time (streaming) and non-real-time data accessible by a client by activating (or "shooting") a media missile. The real-time data (or clocked media data) 22, in a specific example, can refer to media data or media missiles stored or broadcast from Streaming2l's Media Server and Caster or other compatible video server platforms, which are collectively illustrated as data source 30. The non-real-time data is the presentation specification or any other auxiliary information stored on a web server or file server. In the figure, Data Store 40 illustrates generally one or more stores for additional digital data.

As shown in FIG. 1, MRM 20 can output either directly to an SMP 50 or through a web or other front-end interface 55. Also, as shown, MRM 20 acts as a middle manager for communicating media data with SAM 10 and managing data from source 30 and data store 40.

(MRM) 20 also can be understood as representing access to any media data that is available to SAM 10 and SMP 50, such as any authorized and available computer on the Internet.

Synchronized Multimedia Player (SMP) 50 has an internal scheduler to process delay-sensitive data. It can also handle sequencing of static media data such as images

and text. It complies with the SMIL recommended by W3C. Hence, it is able to interact with web servers to retrieve and/or to utilize SMIL presentation specifications. In addition, an SMP can also work with a Media Missile locally on a single platform or in a distributed network environment.

As shown in FIG. 1, SMP 50 can receive data directly from MRM 30, or through interface 55. An SMP 50 can also be directed to receive media data directly from a data store 40.

Media Missile

FIG. 3 illustrates an example media missile container format according to specific embodiments of the invention. This example shows a very specific example bit-wise arrangement of various elements. It will be understood to those of skill in the art from the teachings herein that variations of the field and bit arrangements and formats exist within the scope of the invention. Particularly, logic true may be encoded as a variety of different bit values and field lengths may vary, included using more than one bit for fields indicated as bit fields below. The container format shown in FIG. 3 is further described below.

Media Missile 12 is a document container, which can include active content and passive media content. The active content is machine-aware executable binary or script. Passive content can include a presentation specification, metadata information, and the media data. The media data can be another Media Missile. Thus, Media Missile has a nested structural format. All of the content of a media missile can be stored in a file, directory structure, or database, such as on a server system or computer such as repository 303. In a further embodiment, according to specific implementations, some or all components can be generated on the fly by a multimedia missile delivery system.

The executable binary (or script) 200 and the presentation specification 210 are both optional. If machine-executable code is not present, E-Length 205 is zero. The Continuation Bit 202 tells what to expect for the follow-on Media Missile structure. If Continuation Bit 202 is one and E-Length 205 is zero, this indicates that the entire Media Missile starts from a presentation specification (210). If Continuation Bit 202 is zero and E-Length 205 is zero, it indicates that the entire Media Missile starts from MetaData information 220. Finally, if Continuation Bit 202 is zero and E-Length is "-1", the entire Media Missile consists of pure media data.

Continuation Bit 212 in front of P-Length 215 tells whether the metadata information is included in the media missile (1) or not (0). If no metadata information is available, the data right after Presentation Specification is the media data.

If metadata information 220 is present, the first bit 222 tells whether the data right after the metadata information is another Media Missile (1) or just pure media data (0). In metadata information, the signature field 226 is 128-bits by default to store a digital signature using one-way one-to-one hashing function such as MD4 (Message Digest 4) or MD5 (Message Digest 5). The length of the signature field can also be implicit by the implementation to guard against trivial breakdown and guessing.

Following signature field 226, MetaData 220 holds one or more length field(s) (such as 245 and 295) for media data. Each length field is 64-bit starting with a bit representing URL (1) or payload (0) (such as 242 and 292). In specific embodiments, each length field in MetaData 220 corresponds to a segment of data in media data 230. The actual data stored in 230 can be characters indicating a URL, or can be the media data. Segment(s) of data are stored in sequence according to the sequence of length field(s). Here, URL stands for Universal Resource Locator for World-Wide Web hyperlink; payload indicates media data for the media player. As will be understood in the art, however, URL generally can refer to a locator (such as a filename or database id) for any resource available via a computer or communications system. In specific embodiments, a URL or other locator can indicating streaming or "real-time" data sources, such as sources of live streaming data.

As will be understood from this discussion and other descriptions herein, an important and innovative aspect of the Media Missile Container Specification is that once a user "shoots" the Media Missile from a particular location, from the perspective of the user, the contents of the Media Missile can be delivered apparently from a single location. In contrast, with formats such as a standard HTML or SMIL type page, a client at a user site may have to access dozens of different text, static image, executable, or audio and video files, from dozens of different locations, to complete a presentation. With a Media Missile, a client can access all of the included data from a single location, with the data delivered in the Media Missile format. This aspect also has advantages for data providers in that Media Missile formatted data is easier to manage, distribute, and deliver than is a presentation the requires many different files.

Examples of Authoring Media Missile

What follows are descriptions of the operation of authoring manager 10 according to specific embodiments of the present invention. A wizard, as discussed below, is understood in the art as logic implemented processes that guides a user through a sequence of steps in an assisted and constrained fashion to accomplish a result, such as installing software or creating a particular document or file type. A template, as discussed below, is understood in the art as set of data specifying general format parameters that may be used to determine or guide the final format of one or more final files or data structures. Synchronization primitives are further described below and illustrated in FIG. 6.

Example Basic Authoring Method

The steps to create or update a Media Missile (using Synchronized Authoring Manager (SAM) 10 are as follows.

1. The Synchronized Authoring Manager (SAM) provides template selection for user to create a synchronized multimedia presentation. It can also read from an existing presentation specification for user to update the specification.

2. SAM takes presentation specification and media data as input used to parse and create a tree representation internally. The tree is a top-down representation of the synchronization primitives. Each intermediate node is either a parallel or sequential synchronization primitives. SAM then retrieves or calculates the required fields according to the Media Missile specification for the leaf nodes (individual media) in the tree. This can be a pipeline process.

3. SAM attaches system-required fields, e.g., a 64-bit trailer, to individual media specified in the presentation to become a basic Media Missile with media data only.

4. SAM uses a depth-first tree traversal to calculate and create optional fields based on user input. The optional fields are feasible to apply to any node in the tree depending on user's needs.

5. SAM uses a breadth-first tree traversal to traverse the entire tree and create the final Media Missile.

Sample Sequences of Authoring Interactions

Case 1: Creating Synchronized Multimedia Presentation for a Guided Lecture:

1. A wizard helps author to pick up a lecture template or other template of the presentation skeleton through a point-and-click list, where corresponding layout of the template also displays.

2. The wizard asks the author to pick up a lecture video and corresponding presentation slides in sequence.

3. The author has a choice to use default duration for slide display duration or use MetaData Manager to customize the presentation timeline.

4. If using the MetaData Manager, the wizard displays both the video and corresponding slides at a particular point in time. Author has the liberty to modify the duration setting for each slide.

5. The wizard then prepares the final output, both the presentation specification and media object, in user-specified formats, e.g., SMIL for presentation specification or Media Missile for media object.

6. The wizard asks author whether author wishes to keep the final output locally or upload it to a specific hosting server.

Case 2: Updating Synchronized Multimedia Presentation for a Guided Lecture:

1. Author uses wizard to pick up an existing presentation specification or media missile from a hosting server.

2. The wizard asks the author to delete or add video and slides.

3. The author has a choice to use default duration for slide display duration or use MetaData Manager to customize the presentation timeline similar to step 3 in Case 1.

4. The wizard repeats step 4 and 5 in Case 1.

5. The wizard asks the author to apply the changes and to check if object deleted from the specification should also be permanently removed from the hosting server(s).

6. The wizard repeats step 6 in Case 1.

Example Basic Method for Playing Back Media Missile

Playback of Media Missile according to specific embodiments of the present invention involves three potentially overlapping processes: delivering, assembling, and rendering Media Missiles and their contents.

Example Method for delivering a Media Missile are as follows:

1. User can use a browser to point-and-shoot the Media Missile. Alternatively, user can also use command line interface to launch a local self-contained Media Missile, or run a player with a Media Missile name as input.

2. If the Media Missile contains a complete executable binary (or script), it generally should be able to load itself and run (e.g. it can be loaded by the native operating system of the target device). In specific embodiments, Media Missile takes advantage of the runtime loader characteristics available on most operating system platforms such as UNIX or Windows. For example, runtime executables or scripts are identified by a Magic Number (first two bytes in any file) on UNIX platforms. On Windows platforms, runtime executables and scripts are explicitly identified by a three-letter filename extension registered in the system registry (such as .EXE or .COM). A loader generally also checks to determine if the files are indeed executable. Other operating system platforms may have different mechanisms to load and launch runtime executables that may be used by a media missile.

3. The Media Missile will be streamed and downloaded from an application server or a web server if it is at remote site. A browser for a web or application server is generally able to run or save a remote file. To make Media Missile compatible with these servers, a hyperlink retrieval of a Media Missile in a web-enabled page will implicitly stream only the machine-aware portion of the Media Missile. The machine-aware portion can then be launched and run on a local platform and manage requests for additional portions of the media missile. Because the Media Missile's executable footprint size is known ahead of time, at runtime the executable is able to manipulate the rest of the Media Missile according to its specification.

Example Method for presenting a Media Missile are as follows:

1. The Synchronized Multimedia Player (SMP) can be locally-resident executable code or executable code that is downloaded or streamed with the Media Missile. SMP has a timeline-based scheduler to schedule the playback according to the presentation specification. The presentation specification again may be available within the Media Missile or from another source, such as through web page that is holding the Media Missile.

2. SMP parses the presentation specification to build a presentation tree.

3. SMP fires the presentation from the top of the tree based on the synchronization characteristics (either parallel or sequential) of each intermediate tree node. For parallel nodes, SMP creates independent execution threads to handle each parallel task and each of these threads refers to a global time base. For sequential nodes, SMP can use a single execution thread to step through the sequential tasks.

4. When SMP fires a leaf node, SMP locates and launches the media player and media data.

If the link to the media data in the specification is a relative path, SMP will by default search the Media Missile itself, using the media data's signature or other identifier to see if the media data is available in the Media Missile. This searching can take place on a Media Missile either that exists already at a local machine or on a Media Missile still resident at a remote server. Various protocols are known for a streaming data client to make requests and receive responses from a streaming server and the protocols can be used to search for specific data within the Media Missile. Otherwise, SMP will follow an absolute path either specified in the Media Missile or from the root of the server where the Media Missile or presentation specification was streamed, to locate the media data.

Regarding the player, SMP will look to determine if the player for the particular media data is located on its local machine. If the player is not located on its local machine, SMP will search for the player in the Media Missile or in other data store.

Playing a Media Missile

FIG. 4 illustrates generally the functional elements the enable MM playback according to specific embodiments of the invention. As shown in the figure, MM playback may access media and code data from a variety of locations. A media missile repository 303 can be understood as a primary data storage of media missile data, including passive data and active code data. The repository in specific implementations can be understood to reside on a server, accessible via a local network, wide area network, or the Internet. When residing on a server (as opposed to locally stored), repository can be understood to have capabilities of streaming session servers, including seek to end, search, play and pause capabilities in response to requests from a client. In some embodiments, the repository will generally store active and passive content needed to play a media missile on a variety of platforms, though only content needed for a particular platform may be downloaded from 303 by an SMP.

Local Code Resource 305 can be understood as those code components available at the local device. For some devices, such as a computer workstation on a LAN, the local code resource 305 may be quite extensive. For such devices, a MM scheduler and a variety of players may be permanently or nearly permanently present at the local device and therefore not have to be transferred from a server to play a media missile. For other devices, such as a cellular telephone for example, there might be very little or no local code resource and all code will need to be downloaded from 303.

Web or Application Server Repository 307 can be understood as storage for any active or passive data not in 303 or 305. As described herein, this data may include data referenced by a resource locator (such as a URL) within a Media Missile. An example of such data would be an HTML page or data used to complete a presentation, advertising banners or data from external sources, or any other data referenced in the medial missile or needed to complete playback.

As elsewhere described herein, to begin playing a media missile, an executable binary or script component able to parse a MM is first initiated on a playback device. This executable, as described herein, may be initially loaded from 303, 305 or 307. An initial task, in particular instances, is to parse a presentation specification 402, if present, and then to locate and execute a Real-Time Presentation Scheduler 404. Scheduler 404 then schedules the downloading (if necessary) and initializing of any players 406 needed to playback the actual media data. These players can include a variety of off-the-shelf media players, such as QuickTime™ or RealMedia™ players or the players can include proprietary players. As indicated in the figure, once scheduler 404 has initialized, it schedules the downloading and initialization of any players. Thus, the downloading and initializing of various player and player components is under control of the overall presentation schedule being run by the scheduler.

As shown in the figure, scheduler 404 also determines the ending of a presentation and terminates the necessary components.

The steps to render or playback a Media Missile are as follows:

1. Software (SW) checks if the Media Missile contains an initial executable code portion (binary or script) for the platform. If the checking is positive, go to Step 2. If the checking is negative, goto Step 3. This checking can be accomplished by software

registered at the target device to handle a MM file type or the checking can be self starting if the MM itself is an executable file or script.

2. Since there are machine-aware components in the Media Missile, it can be launched on the fly, or it may be used as an enhancement of the SW already present of the system to render the media data within the Missile.

3. In case there's no machine-aware executable within the Media Missile, the media missile or Operating System will trigger and launch a local SMP with default components to read media data from the Media Missile.

4. SMP handles both barrier and event synchronization with elastic timers for synchronization point.

SMP internally has to handle synchronization events according to the presentation specification 320. Certain specifications define precise timing sequence. Some others have relaxed schedule. Any playback of media(s) can be either sequential or concurrent (parallel).

Sample Sequences of Playback Rendering

Case 3: Playing Back a Synchronized Multimedia Presentation for objects in Media Missile format in a Web Page:

1. Client (SMP) 50 accesses a synchronized multimedia presentation, e.g., formatted in a HTML/XML web page.

2. The names/signatures of the media objects at particular points in time in the presentation timeline are passed to the player(s) in the web page through a player scheduler which is downloaded on the fly or resides on the local computing device.

3. Each player starts to retrieve a media missile header corresponding to the media object name/signature.

4. Each player inspects the header to verify the necessity of downloading and registering embedded lightweight software components in the media missile to render the media data.

5. Each player realizes and loads media-specific rendering components at runtime.

6. For a parallel start of rendering, the scheduler sets a barrier synchronization primitive, further description of which is provided below.

7. Each player is enabled to play back and render media data according to the "Enable" signal from the player scheduler, the starting time, and the duration in the presentation specification.

Case 4: Playing Back a Synchronized Multimedia Presentation for objects in Media Missile format in a distributed environment:

1. A self-contained guidance-free Media Missile, which is downloaded or stored in local storage, is launched. The beginning of the Media Missile is the player scheduler, which can be a script or a binary executable object compatible with the runtime platform. Hence, it would run by itself.

2. The player scheduler then loads a presentation specification, which is attached after any present runtime executable code segment in the Media Missile.

3. The names/signatures of the media objects and their position in the Media Missile at a particular point in time in the presentation timeline are passed to the player(s) specified in the Media Missile through the player scheduler. The player(s) (which can also be a Media Missile by itself) resides in a distributed network environment. Hence, the name of a media object can be a link to a Media Missile over the networks.

4. The procedure then repeats step 3 to 7 in Case 3.

FIG. 5 is a flow chart illustrating an example method for accessing a media missile according to a specific embodiment of the invention.

Playback Timing

To handle both of the sequential or parallel playback of media contents with either precise or elastic timers, an SMP defines the following states: ***started, initialized, played, paused,*** and ***stopped*** for each media player under its control. FIG. 6 illustrates the use of these states with the synchronization primitives and the timing requirements. SMP attached a timer for each synchronization primitive to control the duration. Each media player may also have attached timers to control both delay and duration. Whether the content has additional embedded timing information or not, it can be played back as if it is a continuous media, i.e., media with timing information. Thus, SMP turns some static media types into time-based media such as text or image. The implementation of the synchronization checkpoints in the Real-Time Presentation Scheduler at any point in time is elaborated as follows and as illustrated in FIG. 6.

B1. Sequential Playback with Precise Timer:

In this case, the scheduler 330 pairs the started and the stopped time of any two media players 340. As such, a player may be launched, based on estimated latency, to cache (or buffer) media data before it starts to render the data.

B2. Parallel Playback with Precise Timer:

In this case, the scheduler checks for barrier synchronization on started, initialized, played, and stopped states for all of the players. Each player may be toggled between played and paused states if delay timers are set within the player.

B3. Sequential Playback with Elastic Timer:

In this case, the scheduler pairs the started and the stopped time of any two media players. However, one must be stopped before another is started. Alternatively, it can be a single media player that takes sequential input scheduled in time.

B4. Parallel Playback with Elastic Timer:

In this case, the scheduler only sets barrier synchronization for started and stopped states for all of the players. Each player controls its own status and the scheduler ignores any events from players other than started and stopped events.

As will be understood to those of skill in the art from the teachings herein, the synchronization of multiple players allows the presentation of multi-part media presentation. Case B2, for example, might illustrate a case where a presenter or lecturer is narrating a video clip or a slide-show presentation. The presenter will come to a point in the presentation where it is time to begin the video clip, the clip will begin, and the presenter audio or audio and video, will continue while the video clip proceeds. The video clip might occupy an entire screen space, or the clip might occupy a subwindow in the screenspace, while video of the presenter occupies a different window portion. As shown in FIG. 6, initialization of a Player2 can begin at some point during the activation of Player1. Thus, Player2 or Player3 components need not be downloaded early and might not be downloaded at all if a user halts ends access to a presentation. It will be understood from the teachings herein, as discussed with respect to FIG. 3, that the format of the media missile allows a multi-player presentation to continue from a media missile location, downloading lightweight software components as needed according to different timer strategies as shown in FIG. 4.

Example Application (WebAds)

Further understanding of aspects of the present invention will be provided by considering the following example. This specific example is provided to illustrate various aspects of the present invention, but the invention should not be limited by any of the details provided herein. This example involves use of the herein described media

missile format, according to specific embodiments of the present invention, to enable a WebAds streaming presentation.

Using A Packaging Utility to Package a MM

As a simple example, a MM can be created from an existing HTML presentation as described below. In a particular embodiment, a package utility may be launched as follows:

package webAdsUrl [webAdsDir startingHtmFile]

In this example, *webAdsUrl* indicates a URL pointing to an external HTML location, in case the remainder of the packaged file is corrupted or empty when the MM is accessed. *webAdsDir* indicates an absolute or relative path to specify a directory the includes files to be placed in the MM and *startingHtmFile* indicates an initial html file that will be used as the initial outline of an MM. As an example, consider a Web Ads directory that includes the following files:

c:\webAds\s21WebAds.htm
c:\webAds\otherLinks.htm
c:\webAds\logo.gif
c:\webAds\sponsor\abcCo.jpg

In this example, the package utility can be run with a command line such as:

c:\s21\bin\package "http://www.abcCo.com" c:\webAds s21WebAds.htm

This example can be further understood with reference to FIG. 7 to FIG. 13. FIG. 7 shows a standard browser display of example source files that may be packaged into a media missile in an example in accordance with specific embodiments of the present invention. According to various specific embodiments of the present invention, and as will be understood from the teachings provided herein, this source data file could be any type of media data. Here, the example media data includes an HTML file and its included image files, including a GIF and a JPG file. The HTML file also includes a reference to another HTML file, otherLinks.htm, which could include any additional links or html-type data.

FIG. 8 shows an example interface for packaging source data into a media missile according to specific embodiments of the present invention. A variety of different packaging tools can be used in different embodiments of the invention, including advanced wizards for creating MM as described herein. In this example, however, a

simple packaging tool is used to create WebAds, and this packaging tool requests just the four parameters illustrated.

When playing such a file, a player can extract the packaged file and place it into a temp directory. If the file is an audio-only clip, a default HTML browser page can be displayed, or the starting html file (e.g. s21WebAds.htm, as shown in FIG. 13) can be displayed in the player window. When the player quits, it generally will remove the temporary files and temporary directory.

If player.dat is missing, a player can use a centrally located file (such as www.Streaming21.com/ads) or show error and then stop the application. If player.dat is corrupted or player.dat contains no files, a player will use the default URL inside the player.dat, if possible. In the worst case, the player can display the default bitmap for Audio playback.

FIG. 9 illustrates an example general metadata only media missile format according to specific embodiments of the invention. In this example, the file begins with metadata information. As indicated in the figure, the metadata includes one or more URLs for associated files with the streaming presentation. Following the metaData is the mediaDataBlock, which contains the actual media object data or locators for media object data.

FIG. 10 illustrates an example metadata only media missile data structure according to specific embodiments of the invention. Note that in this example, the values shown at the top of the figure are assumed. Note that as indicated in the figure, all of the source files are incorporated into the media data area of the media missile file, with the metadata area indicating the identity of the portions of data in the media data area.

FIG. 11 illustrates an example media missile data structure according to specific embodiments of the invention. Comparing this figure with FIG. 3 will show one specific data structure example of how a Media Missile according to specific embodiments of the present invention may be realized. Note that objects in

FIG. 12 illustrates an example binary encoded metadata only media missile data according to specific embodiments of the invention.

Optional processing for WebAd in a media missile allows ignoring an Executable Binary of Script portion and a Presentation Specification portion, if either are found in the

media missile as, for example, shown in FIG. 11. . An example of seeking in a media missile file to skip this data is as follows:

```
OpenFile
Seek to ending-8bytes
BOOL contination_bit = readlBit();
__int64 eLength = readNextBits(63);
SeekBackToBeginning;
if (eLength==-1)
     error                          // required metaDataBlock is absent
else if (eLength)
     skipFirstBytes(eLength);       // skip Executable Block
if (p_blk_exist)
{ BOOL next_blk_is_meta_data = readlBit();
  if (!next_blk_is_meta_data) error; // required metaDataBlock is absent
  else
  {__int64 p_blk_len = readNextBits(63);
   skipFurtherBytes(p_blk_len);  // skip Presentation Block
  }
}                                 // Reach Meta Data
```

FIG. 13 illustrates an example presentation of a media missile in a player according to specific embodiments of the present invention. In a further embodiment, a player can use a local *ads.html* page by default, instead of always needing to contact a server for the html page. The local *ads.html* can include references to other files, such as avi, gif, jpeg, html or similar files. In specific embodiments, these may be restricted to the same directory in which *ads.html* resides and its sub-directory. *Ads.html* and related files can be packaged into a media missile format, with, as an example, a name in the form 'player.dat'.

EXAMPLE SYNCHRONIZED AUTHORING MANAGER IMPLEMENTATION

An SAM according to specific embodiments of the invention provides a user-friendly GUI that allows an author to successfully complete a Media Missile (or, in some embodiments, just a well-formed Presentation Specification) by applying mouse clicks and keystrokes, with no programming background required. SAM provides a wizard-driven guide to help non-programming users generate multimedia presentations. SAM wizards generate MM based on a series of templates, so that the multimedia presentations created through SAM are well-formed and can be parsed and rendered by any compliant player. In further embodiments, SAM provides a tool to preview different media such as text, images, html files, audio and video clips and facilitates the generation of synchronization data between different media. SAM can be implemented in Java or other programming environments. FIG. 14 illustrates an example general method for guiding a

user in creating a media missile or presentation data according to specific embodiments of the invention.

Template Support

A SAM according to the present invention provides a set of different standard templates to allow authors to create presentations such as video lectures, headline news, and slide shows.

For example, a video lecture template makes it easy for an author to generate a video lecture (video synchronized with HTML slides).

A headline news template is designed for virtual briefings (video synchronized with text streams at bottom) in scenes such as: headline news -- video with anchorperson and text containing headlines; online event -- video with event introduction and text containing registration information; product advertisement -- video with product advertisement and text containing pricing information.

A slide show template is designed for illustrated audio (audio synchronized with images) in scenes such as: guided tour -- narrated map with scenic photos to guide users through a site tour; process presentation -- audio commentary to slides demonstration of a process; or online publication -- sound book or magazine with reference materials in pictures.

While some prior multimedia authoring tools supported only non-streamed data types, an SAM according to the present invention, allows inclusion of streaming media along with other media types. FIG. 15 illustrates an example of a Synchronized Authoring Manager (SAM), also referred to as a Synchronized Authoring Tool (SAT), according to specific embodiments of the invention.

SAM Core Components

In specific embodiments, SAM includes four components: Presentation Wizard, Preview Manager, Presentation Specification Data Manager, and Media Missile Generator.

Presentation Wizard is SAM's kernel that provides an easy way to enhance disparate media into integrated and dynamic multimedia content. It guides users to create presentations in just a few steps.

Preview Manager can display any allowed medium element. If the medium is time-irrelevant, such as an image, a text stream, or an html file, then the exact content

will be shown in the preview window. If the medium is time-intrinsic, such as an audio or video clip, then it will be played continuously from the beginning in the preview window. If the medium is from a live streaming source, such as a live camera or video feed, a live sample of the medium can be played in the preview window. The Preview Manager can be activated inside and outside the Presentation Wizard. Users can use the Preview Manager to preview the medium before incorporating it into a multimedia presentation.

Presentation Specification Data Manager is responsible for creating Presentation Specification Data (referred to at times in this context as metadata, although herein it refers to data for the presentation specification and not the Media Missile MetaData described above) information and a presentation specification, which is used to synchronize different media within a multimedia presentation. Users can playback an audio or video clip or stream, meanwhile recording relevant timing information for static media based on the content of the clips. For example, a user wishing to create a presentation that displays a narrated map (using audio), showing some slides of scenic photos (using images), and access from a live outdoor camera, indicating current weather. With Presentation Specification Data Manager, the user can decide the timing point to show the next slide according to the audio content or to open a window to display the streaming (or live) data. The information is saved in a data structure and used later for creating the corresponding presentation content.

Document Generator features automatic creation of a presentation specification and/or a Media Missile. The document's structure is based on one of the templates, and the media and timing information come from Presentation Wizard operation. The Document Generator can package the various media data with the presentation data to create and any necessary or executable content to create a Media Missile. As an option, the Document Generator can also output just an SMIL-type presentation specification.

Creating a Media Missile Using SAM

Step 1 - Choosing A Template

At first, users choose the type of presentation they wish to create. Each template is designed with a specific purpose in mind. SAM can support templates such as:

Video Lecture Template

Video lecture usually consists of a video clip or a streaming feed from a streaming data source such as a live camera or continuously encoded television broadcast,

synchronized with HTML slides. The video can be any media format supported by a target Player. The HTML slides are any conforming HTML files or other files that can be handled by a player. In some embodiments, if the slides are originally in another format (such as Microsoft PowerPoint) they are first converted to HTML format using a "Save as HTML" command in PowerPoint.

Headline News Template

Headline news usually consists of a video clip or a streaming feed synchronized with text streams. The video clip can be any media format supported by a target player. Text streams are generally formatted ASCII text. In a specific embodiment, users can directly type any text they want in the second step.

Slide Show Template

Slide show usually consists of an audio clip or streaming audio feed synchronized with image slides. The audio clip can be any media format supported by a target Player. The image slides are image files in GIF, JPEG, or any supported format.

Step 2 - Adding and Previewing Media (Preview Manager)

The second step is to add media for the presentation. Users can activate the Open File dialog box to select the files or streaming sources they wish to add. Users can also activate the Preview Manager to preview any selected medium. In various situations, users can decide how to create presentation data by using either the Presentation Specification Data Manager or SAM's default setting. If the Presentation Specification Data Manager is chosen, then the wizard will activate the Presentation Specification Data Manager as the third step. If SAM's default setting is chosen, then SAM assumes, for example, that slides are displayed sequentially, each slide has the same duration and the first slide starts displaying at the very beginning of the presentation. Therefore, SAM can calculate the presentation specification Presentation Specification Data automatically and does not need to activate the Presentation Specification Data Manager.

Step 3 - Creating Presentation Specification Data (Presentation Specification Data Manager)

In this step, the Presentation Specification Data Manger is activated to create the presentation specification Presentation Specification Data. Presentation specification Presentation Specification Data is used to synchronize different media within a multimedia presentation. It is as simple as timing information when to change to the next slide (image, text, or html file) or when to open a window for live media stream, possibly

synchronized with the content of time-intrinsic media (such as video or audio clips). With Presentation Specification Data Manager, users can decide at which timing point a slide or window starts displaying based on the timeline of another playing audio or video clip. An example implementation of a video lecture, in which two slide files (formatted as HTML files) and a opening of a live streaming source are synchronized to an explanatory video clip based on the Presentation Specification Data file shown below:

sld001.htm>1231646556
sld002.htm>5464312313
livesource.s21>7634537334

In the Presentation Specification Data file, the string before character '>' is the indicator or handle of one slide file or a streaming source, while the string after character '>' is the exact media time when the slide or streaming source should start displaying. Whenever the current media time exceeds one of the media times in the Presentation Specification Data file, the corresponding displayed is activated. Since the media times are associated with the timeline of one video clip, then the synchronization between the video and the slide files is implemented.

Step 4 - Generating Media Missile Presentation (Document Generator)

Finally, the wizard shows the summary information such as the template name of the presentation, file names or URLs of selected media, and the Presentation Specification Data for synchronization. After clicking Finish button, the Presentation Wizard sends all the information to a Document Generator. The Document Generator creates a SMIL presentation specification or MM document based on a selected template file. In one embodiment, it reads the template file line by line, parsing it, and collecting and then placing the corresponding media data, file names, URLs or presentation data into a Media Missile structure.

SAM Component and Screen Implementation Examples

FIG. 16 illustrates an example of a Preview Manager GUI according to specific embodiments of the present invention. This Preview Manager is a single window, split into a media preview area and a control area at the top of the window. A text field control contains the current medium file location. Users can choose a medium in any way know for selecting files or indicating locations or streaming sources (URLs) such as typing a new path inside the text field; or pressing the button labeled ... to select a location. The

selected path is then shown in the text field. After selection, users can press Preview button to start displaying the medium.

According to specific embodiments, each Presentation Wizard step can be implemented as a dialog box. Each dialog box has three buttons in common. Users can click Back button to return to the previous wizard step or click Next button to go forward to the next wizard step. Users can click Cancel button to close the wizard at any step without generating a document.

FIG. 17A-C illustrate examples of choosing various presentation templates according to specific embodiments of the present invention.

The Adding and Previewing Media functionality can change based on the template selected. FIG. 18A shows the Video Lecture template Add and Preview Media screen. Labels such as "Add Video Clip", "Add Slide Files", "Video Duration", and "Each Slide Duration" are set based on the media support of this template. Users can add, remove, or preview media in this dialog box. Users can click Remove button to delete selected slide file in the list box. Users can also click Preview button to activate the Preview Manager for playing back the selected medium file. Figure 7-19 shows a selected HTML slides displayed in the Preview Manager.

FIG. 19A illustrates the situation when Headline News template is selected in step one. Headline News consists of video synchronized with text notations, which come from direct input instead of text file. Therefore, a dialog box FIG. 19B is provided for direct text input and is activated when Add Text Notations button is clicked.

Based on the template selection, GUIs can also present different Presentation Specification Data creation methods. FIG. 19A, for example, shows "Using Presentation Specification Data Manager" radio button, which means the wizard will activate the Presentation Specification Data Manager dialog box as the third step. "Using Default Setting" radio button indicates that users just need to type in the duration data. The wizard assumes that each slide has the same duration and the first slide starts displaying at the very beginning of the presentation. Therefore, the wizard does not need to activate the Presentation Specification Data Manager and goes directly to the last step – step 4.

Similarly to Step 2, the functionality of the Presentation Specification Data Manager changes based on the template selection in step one. FIG. 20A shows an example when users select the Video Lecture template in step one and choose some

media files in step two. The "Video Lecture" label is shown in the Presentation Specification Data Manager. Labels such as "Video File", "HTML Slide List", "HTML Slide Preview", and "HTML Slide File Name" are set based on the media support of Video Lecture template. The left part of the Presentation Specification Data Manager is the video monitor area. The file path is shown in the Video File text field, and the video is rendered and played by a Preview module. When the video starts playing, the MediaProgressMonitor activated in step two updates the video Current Time every 50 milliseconds.

The right part of the dialog box is the slide timeline setting area. Each slide is not only included in the HTML Slide List box, but also shown in the Slide Timeline Recording panel in order. Users can click Reset button to reset each TimeLine text field in the panel to 0.0 second and click Sort TimeLine button to sort the slides in timeline order. FIG. 20B shows the situation when users select the Headline News template in step one.

As shown in Figure FIG. 20A, Slide Timeline Recording panel in Presentation Specification Data Manager is the GUI part dynamically generated according to the users' selection in step one and two. Each slide file corresponds to one small child panel that consists of a File Name text field, a Record button, and a TimeLine text field. Since the video timeline is the entire video lecture presentation timeline, users can click Record button to set the starting time of the corresponding slide based on the video content. When one Record button is clicked, the exact time is shown in the corresponding TimeLine text field and the following slide is shown in the HTML Slide Preview area.

As a further option, according to specific embodiments, when Users click the Sort button, the slides in the Slide Timeline Recording Panel are sorted in timeline order, which means the slide with the earliest starting time shows in the first line of the panel and so on. Bubble sort is simple and effective; therefore, it can be used as a sort algorithm for the timeline.

Because generating presentation is the last step in the wizard guide, presentation information such as template name, selected media files, automatically generated or manually generated (by Presentation Specification Data Manager) Presentation Specification Data is summarized in the Presentation Information text area. Meanwhile users can define the presentation title, author name, and copyright. When Finish button is

clicked, the Document Generator will get all the information and generate a presentation independently. FIG. 21 illustrates an example last step of a presentation generation.

SAM can utilize live multimedia streams by using Real-time Transport Protocol (RTP) for streaming media objects, and Real-time Transport Control Protocol (RTCP) for controlling and managing remote stream objects like a traditional VCR does (play, stop, and pause). RTP is the Internet standard for the transport of real-time data including audio and video stream. RTP can be used for media-on-demand application as well as interactive services such as Internet telephony. RTCP is the control protocol designed to work in conjunction with RTP.

Further Example Source File

Below is one example of a SMIL-type presentation specification file that could be part of a media missile according to specific embodiments of the present invention and could be accessed through a synchronized media player according to specific embodiments of the present invention. To display this file on a target machine, for example, would require players such as video, audio, HTML text, and image players, each of which could be resident at a target machine or packaged in a media missile.

```
<smil>
  <head>
   <meta name="author" content="S. Shim" />
   <meta name="copyright" content="SJSU" />
   <layout>
     <root-layout    background-color="192,192,192"          width="1200"
              height="620"/>
     <region    id="videoregion"    left="40"    top="10"    width="150"
              height="100"              z-index="0"         background-
              color="192,192,192"/>
     <region    id="lectureregion"    left="10"    top="220"    width="450"
              height="200"              z-index="0"         background-
              color="192,192,192"/>
     <region    id="slideregion"    left="240"    top="10"    width="250"
              height="200"              z-index="0"         background-
              color="192,192,192"/>
 </layout>
  </head>
  <body>
   <par>
     <video src="demo\gm.mpg"  region="videoregion"/>
     <seq>
      <html        src="demo\car1.gif"        begin="5s"        dur="3s"
              region="slideregion"/>
      <html src="demo\car2.gif" dur="5s" region="slideregion"/>
      <html src="demo\car3.gif" dur="5s" region="slideregion"/>
      <html src="demo\car6.gif" dur="5s" region="slideregion"/>
      <html src="demo\car5.gif" dur="5s" region="slideregion"/>
      <html src="demo\car4.gif" dur="5s" region="slideregion"/>
      <html src="demo\car5.gif" dur="5s" region="slideregion"/>
     </seq>
```

```
        <seq>
         <html          src="demo\gm1.html"          begin="6s"          dur="6s"
                region="lectureregion"/>
         <html src="demo\gm2.html" dur="6s" region="lectureregion"/>
         <html src="demo\gm3.html" dur="6s" region="lectureregion"/>
         <html src="demo\gm4.html" dur="6s" region="lectureregion"/>
         <html src="demo\gm5.html" dur="6s" region="lectureregion"/>
        </seq>
        </par>
    </body>
</smil>
```

Embodiment in a Programmed Information Appliance

FIG. 22 is a block diagram showing a representative example logic device in which aspects of the present invention may be embodied. The invention can be implemented in hardware and/or software. In some embodiments of the invention, different aspects of the invention can be implemented in either client-side logic or a server-side logic. As will be understood in the art, the invention or components thereof may be embodied in a fixed media program component containing logic instructions and/or data that when loaded into an appropriately configured computing device cause that device to perform according to the invention. As will be understood in the art, a fixed media program may be delivered to a user on a fixed media for loading in a users computer or a fixed media program can reside on a remote server that a user accesses through a communication medium in order to download a program component.

FIG. 22 shows an information appliance (or digital device) 700 that may be understood as a logical apparatus that can read instructions from media 717 and/or network port 719. Apparatus 700 can thereafter use those instructions to direct server or client logic, as understood in the art, to embody aspects of the invention. One type of logical apparatus that may embody the invention is a computer system as illustrated in 700, containing CPU 707, optional input devices 709 and 711, disk drives 715 and optional monitor 705. Fixed media 717 may be used to program such a system and may represent a disk-type optical or magnetic media, magnetic tape, solid state memory, etc.. The invention may be embodied in whole or in part as software recorded on this fixed media. Communication port 719 may also be used to initially receive instructions that are used to program such a system and may represent any type of communication connection.

The invention also may be embodied in whole or in part within the circuitry of an application specific integrated circuit (ASIC) or a programmable logic device (PLD). In

such a case, the invention may be embodied in a computer understandable descriptor language which may be used to create an ASIC or PLD that operates as herein described.

Other Embodiments

The invention has now been described with reference to specific embodiments. Other embodiments will be apparent to those of skill in the art. In particular, a user digital information appliance has generally been illustrated as a personal computer. However, the digital computing device is meant to be any device for interacting with a remote data application, and could include such devices as a digitally enabled television, cell phone, personal digital assistant, etc.

While the forgoing and attached are illustrative of various aspects/embodiments of the present invention, the disclosure of specific sequence/steps and the inclusion of specifics with regard to broader methods and systems are not intended to limit the scope of the invention which finds itself in the various permutations of the features disclosed and described herein as conveyed to one of skill in the art.

Furthermore, while the invention has in some instances been described in terms of client/server application environments, this is not intended to limit the invention to only those logic environments described as client/server. As used herein, "client" is intended to be understood broadly to comprise any logic used to access data from a remote system and "server" is intended to be understood broadly to comprise any logic used to provide data to a remote system.

It is understood that the examples and embodiments described herein are for illustrative purposes only and that various modifications or changes in light thereof will be suggested by the teachings herein to persons skilled in the art and are to be included within the spirit and purview of this application and scope of the claims.

WHAT IS CLAIMED:

1. A method of accessing streaming media content comprising:
initiating a media missile;
executing a scheduler, said scheduler loading a presentation specification;
under control of said scheduler, initiating one or more players;
passing indications of media objects in said media missile to one or more players; and
at said one or more players, under control of said scheduler, presenting said media objects from said media missile.

2. The method of claim 1 further comprising:
at said one or more players, retrieving one or more headers from said media missile corresponding to a media object indicated for said one or more players;
at said one or more players, inspecting said one or more headers to determine the necessity of downloading lightweight software components from said media missile to render the media data; and
at said one or more players, loading media-specific components at runtime if needed to access said media data.

3. The method of claim 1 wherein said accessing is performed on a mobile platform with limited pre-installed software capability.

4. The method of claim 1 wherein said accessing is performed on a mobile platform with limited memory footprint.

5. The method of claim 1 wherein said accessing enables synchronized presentation on a mobile platform.

6. The method of claim 1 further comprising:
at said one or more players, setting one or more barrier synchronization primitives.

7. The method of claim 1 wherein said one or more players plays media data according to an enable signal from said scheduler.

8. The method of claim 1 wherein said initiating comprises launching a media missile from local storage, said media missile including a initial player scheduler able to run on a local device's runtime platform.

9. The method of claim 1 wherein said initiating comprises launching a media missile from remote storage, said media missile including a initial player able to run on a local device's runtime platform.

10. The method of claim 1 wherein said initiating comprises accessing a presentation formatted in a web page.

11. The method of claim 1 wherein said scheduler is taken from said media missile.

12. The method of claim 1 wherein said scheduler is preinstalled on an access device.

13. The method of claim 1 wherein said presentation specification is loaded from said media missile.

14. The method of claim 1 wherein one or more of said indications of media objects includes a signature.

15. The method of claim 1 wherein said media missile is a document container including active content and passive content, said active content comprising machine-aware executable components and said passive content comprising presentation specification, metadata information, and media data.

16. The method of claim 1 wherein said media data can include another media missile.

17. The method of claim 15 wherein said active and passive content of said media missile can be accessed from a single streaming file location.

18. A method of accessing streaming media content comprising:

initiating a media missile;

wherein said media missile is a document container capable of holding machine-aware active content and passive content in a variety of formats; and

passing indications of media objects in said media missile to one or more players.

19. The method of claim 18 wherein said initiating comprises:
using a browser to point-and-shoot said media missile.

20. The method of claim 18 wherein said initiating comprises:
using a command line interface to launch a local self-contained media missile.

21. The method of claim 18 wherein said initiating comprises:
initiating a player with media missile name as input.

22. The method of claim 18 wherein said initiating comprises:
wherein said media missile contains a complete executable component and is able to load itself and run in accordance with runtime loader characteristics available on a particular operating system platform.

23. The method of claim 18 wherein at least one of said one or more players is included in said media missile.

24. The method of claim 18 wherein at least one of said one or more players is an external player.

25. The method of claim 18 wherein said external player is an MPEG player.

26. The method of claim 18 wherein a hyperlink retrieval of said media missile in a web-enabled page implicitly streams only an initial active machine-aware portion of said media missile with the remainder of said media missile streamed under control of said initial active machine-aware portion.

27. The method of claim 18 wherein said accessing may be performed on a platform with limited pre-installed software capability.

28. The method of claim 18 wherein said accessing is performed on a platform with limited memory footprint.

29. The method of claim 18 wherein playback of a Media Missile further comprises potentially overlapping processes of delivering, assembling, and rendering media missiles and their contents.

30. A method of playing a synchronized multimedia presentation comprising:
initiating a media missile;
wherein said media missile is a document container capable of holding machine-aware active content and passive content in a variety of formats;
initializing a synchronized multimedia player with a timeline-based scheduler to schedule playback according to a presentation specification; and
passing indications of media objects in said media missile to one or more players.

31. The method of claim 30 wherein said presentation specification is available through a web page.

32. The method of claim 30 wherein said presentation specification is available from said media missile.

33. The method of claim 30 wherein said synchronized multimedia player is resident on a platform prior to said initiating.

34. The method of claim 30 wherein said synchronized multimedia player is in said active content of said media missile.

35. The method of claim 30 further comprising:
parsing said presentation specification to build a presentation tree; and
firing said playing from the top of said tree based on the synchronization characteristics of intermediate tree nodes.

36. The method of claim 35 further comprising:
for one or more parallel nodes, creating independent execution threads to handle each parallel task, each independent thread referring to a global time base.

37. The method of claim 35 further comprising:

for one or more sequential nodes, using a single execution thread to step through sequential tasks.

38. The method of claim 35 further comprising:
upon firing a leaf node, locating a correct media player and media data.

39. The method of claim 30 further comprising:
if a link to media data in said presentation specification is a relative path, searching said media missile for said media data; and
if a link to media data in the specification is not a relative path, following an absolute path to locate the media data.

40. The method of claim 39 wherein said absolute path is either specified entirely in said link or is initiated from a root of a server from where said media missile was streamed.

41. A method of accessing a multimedia presentation comprising:
checking a media missile to determine if said media missile contains an executable portion for a current platform;
if said checking is positive, launching machine-aware components in said media missile on the fly or as an enhancement of local software to render media data within said media missile; and
if said checking is negative, launching a local media player with default components to read media data from said media missile; wherein said media missile is a document container capable of holding machine-aware active content and passive content in a variety of formats.

42. The method of claim 41 further comprising:
initializing a synchronized multimedia player with a timeline-based scheduler to schedule playback according to a presentation specification; and
passing indications of media objects in said media missile to one or more players.

43. The method of claim 42 wherein said player handles both barrier and event synchronization with elastic timers.

44. The method of claim 42 wherein said player internally handles synchronization events according to a presentation specification.

45. The method of claim 44 wherein a presentation specification can define precise timing sequence or a relaxed schedule.

46. The method of claim 44 wherein a playback can be either sequential or concurrent (parallel).

47. A media missile streaming document container comprising:

a first location for active media content; and

a second location for passive media content;

wherein said active media content comprises machine-aware executable code and said passive media content can comprise presentation specification, metadata information, and media data.

48. A streamable media missile file comprising:

a first portion for holding executable content;

a second portion for holding a presentation specification;

a third portion for holding metadata; and

a fourth portion for holding one or more media data objects.

49. The device according to claim 47 wherein one or more of said media data objects can include another media missile, providing a nested format.

50. The device according to claim 47 wherein said media missile file can be managed at a server and distributed from a server as a single file.

51. The device according to claim 47 wherein said media missile file is generated on the fly at a server though is delivered to a user as a single streaming file.

52. The device according to claim 47 further comprising:

a first length field indicating if machine-executable code is present and the length of said code if present; and

a first continuation field indicating presence of other portions in said media missile file.

53. The device according to claim 52 wherein:

if said first continuation field is logical true and said first length field indicates a zero length, said media missile begins with said presentation specification;

if said first continuation field is logical false and said first length field indicates a zero length, said media missile begins with said metadata; and

if said first continuation field is logical false and said first length field indicates a negative value, said media missile comprises pure media data.

54. The device according to claim 47 further comprising:

a second length field indicating if a presentation specification is present and the length of said presentation specification if present; and

a second continuation field indicating presence of other portions in said media missile file.

55. The device according to claim 54 wherein:

if said second continuation bit field is logical true, then metadata information follows said presentation specification; and

if said second continuation bit field is logical false, then media data follows said presentation specification.

56. The device according to claim 54 further comprising:

a third length field indicating if metadata is present and the length of said metadata if present; and

a third continuation field indicating whether data after said metadata is another media missile or media data.

57. The device according to claim 56 wherein:

said metadata comprises one or more media object length fields indicating length of one or more media data objects in said media missile.

58. The device according to claim 57 wherein:

wherein said one or more media objects are stored in sequence according to a sequence of said media data length field(s).

59. The device according to claim 56 wherein one or more media data length fields includes a field indicating whether a corresponding object is a resource locator or is a payload.

60. A method of assisting a user in creating a synchronized multimedia presentation comprising:

prompting said user to select a template for said presentation;

prompting said user to select one or more media data objects;

prompting said user to select an option for generating presentation specification data; and

preparing final output, including a presentation specification.

61. The method according to claim 60 wherein said final output comprises a multimedia missile that can include media objects, a presentation specification, metadata, and active components in a container deliverable as a stream from a single streaming source location.

62. The method according to claim 60 wherein said final output comprises a presentation specification in a standard descriptive tagged language.

63. The method according to claim 60 wherein said method enforces generation of a well-formed synchronized multimedia presentation.

64. The method according to claim 60 wherein final output can include both precise and elastic synchronization primitives.

65. The method according to claim 60 further comprising:

prompting said user to modify duration settings for media data objects; and

prompting said user to use a presentation metadata manager to customize a presentation timeline.

66. The method according to claim 65 further comprising:

displaying the video and corresponding slides at a particular point to an user to assist creating a presentation.

67. The method according to claim 61 further comprising:

prompting said user to select whether to keep the final output locally or upload it to a specific hosting server.

68. The method according to claim 60 further comprising:

accepting presentation specification and media data as input to said authoring method.

69. The method according to claim 60 further comprising:

creating a top-down tree representation of synchronization primitives wherein each intermediate node is either a parallel or sequential synchronization primitive; and

determining required fields according to a media missile specification for terminal leaf nodes in said tree.

70. The method according to claim 69 further comprising:

attaching fields to individual media data specified in a presentation so that said individual media have the format of a media missile with media data only.

71. The method according to claim 70 further comprising:

using a depth-first tree traversal to calculate and create optional fields based on user input; and

using a breadth-first tree traversal to traverse the entire tree and create the final media missile.

72. A system for streaming synchronized multimedia presentations comprising:

a real-time data source providing a plurality of streaming media objects;

a data store for holding and retrieving presentation specifications or media missiles; and

a repository manager that transmit a streaming media missile to a target device, where the media missile can include packaged indications from said real-time data source, can include static media objects, and can include presentation specification,

73. The system of claim 72 further comprising:

an authoring manager for receiving authoring specifications for a streaming presentation.

74. The system of claim 72 wherein said authoring manager accepts indications provided by an author and using said indications prepares a well-formed streaming presentation in accordance with a predefined descriptive tagged language.

75. The system of claim 72 wherein said repository manager handles transmission of executable components as said components are needed and are requested by a synchronized multimedia player said for accessing a portion of said streaming presentation.

76. The system of claim 75 wherein said repository manager and said player are able to begin pre-fetching delay-sensitive media data.

77. The system of claim 72 wherein data is delivered from said repository manager in a container that provides indications for machine-aware executable code, media data, and associated information.

78. The system of claim 77 wherein said container can include one or more additional containers in a nested structure.

79. The system of claim 77 wherein said container holds all active and passive content necessary to play a synchronized multimedia presentation.

80. The system of claim 72 wherein said authoring manager understands various language specifications, formats, and templates in synchronized multimedia.

81. The system of claim 72 wherein said authoring manager includes a wizard that guides a user to compose a synchronized presentation and retrieve media data from a media repository manager to produce a media missile.

82. The system of claim 72 further comprising a synchronized multimedia player including an internal scheduler to process delay-sensitive data.

83. The system of claim 82 wherein said synchronized multimedia player can present presentations in accordance with SMIL presentation data.

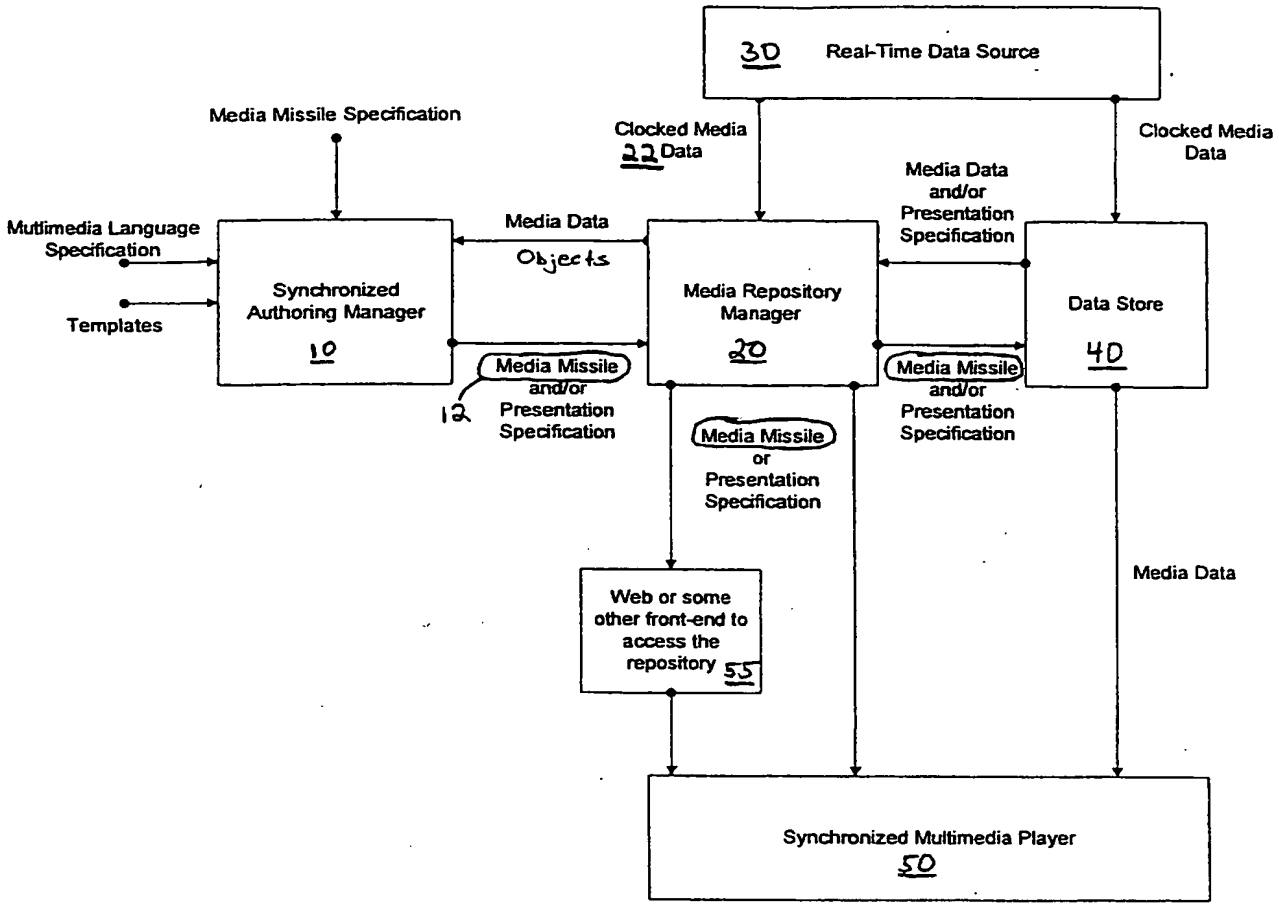

30 Real-Time Data Source
Media Missile Specification
Clocked Media 22 Data
Clocked Media Data
Media Data and/or Presentation Specification
Mutlimedia Language Specification
Templates
Synchronized Authoring Manager 10
Media Data Objects
Media Repository Manager 20
Data Store 40
Media Missile and/or Presentation Specification 12
Media Missile and/or Presentation Specification
Media Missile or Presentation Specification
Media Data
Web or some other front-end to access the repository 55
Synchronized Multimedia Player 50

FIG. 1

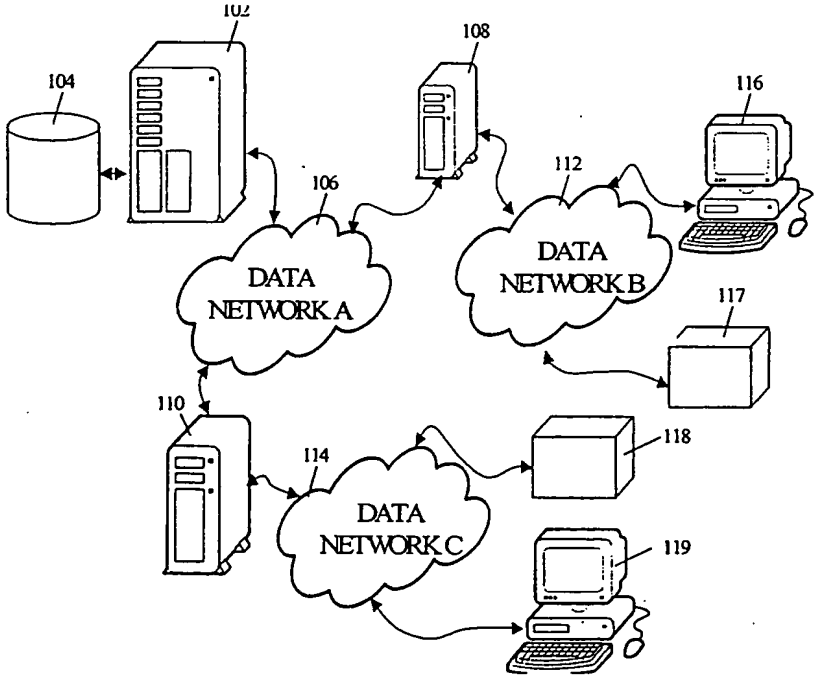

102
104
108
116
106
112
DATA NETWORK A
DATA NETWORK B
117
110
114
118
DATA NETWORK C
119

FIG. 2

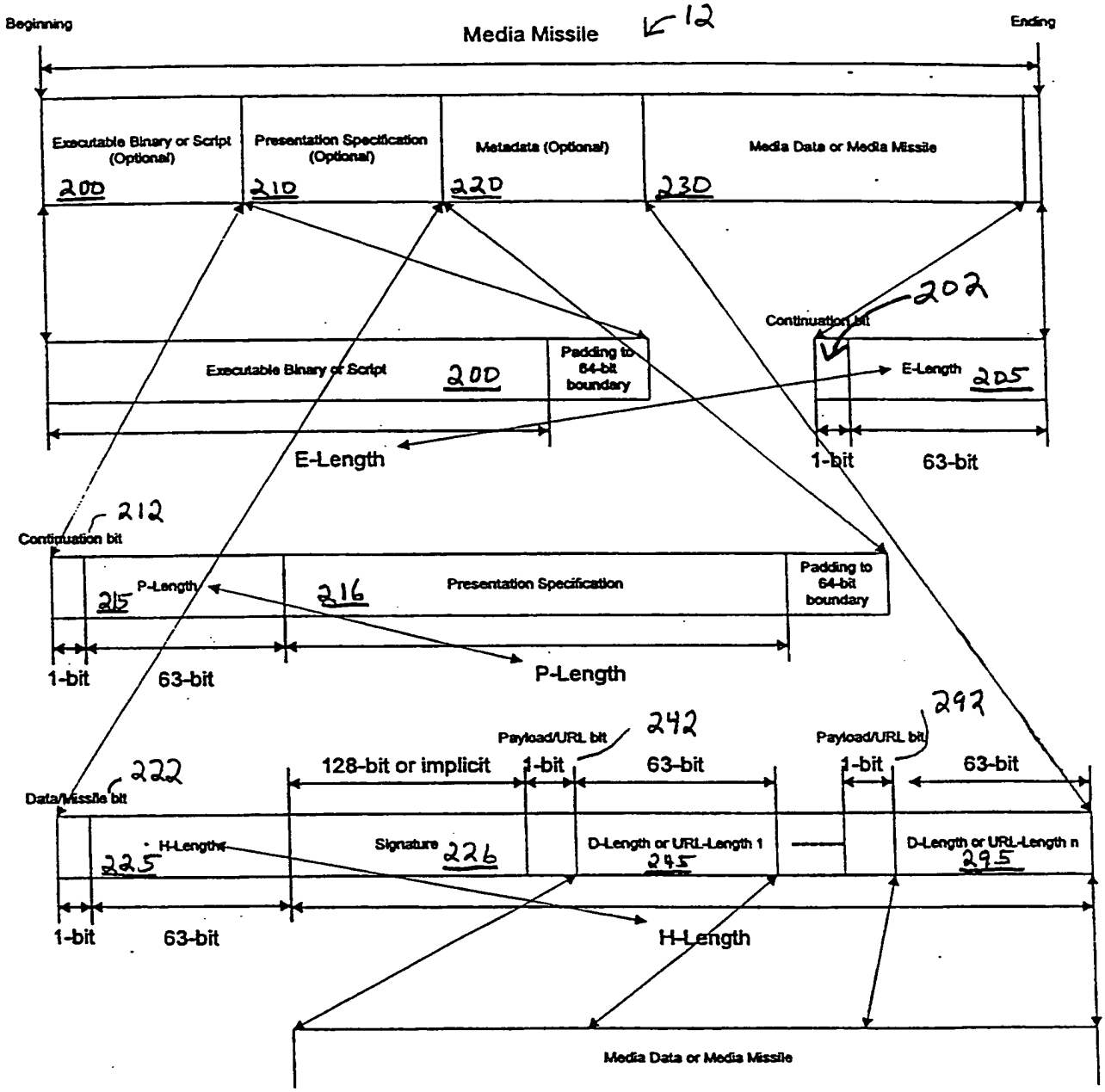

Beginning
Media Missile
12
Ending
Executable Binary or Script (Optional)
200
Presentation Specification (Optional)
210
Metadata (Optional)
220
Media Data or Media Missile
230
202
Continuation bit
Executable Binary or Script
200
Padding to 64-bit boundary
E-Length
205
E-Length
1-bit
63-bit
212
Continuation bit
215
P-Length
216
Presentation Specification
Padding to 64-bit boundary
1-bit
63-bit
P-Length
242
Payload/URL bit
128-bit or implicit
1-bit
63-bit
297
Payload/URL bit
1-bit
63-bit
222
Data/Missile bit
225
H-Length
Signature
226
D-Length or URL-Length 1
245
D-Length or URL-Length n
295
1-bit
63-bit
H-Length
Media Data or Media Missile

FIG. 3

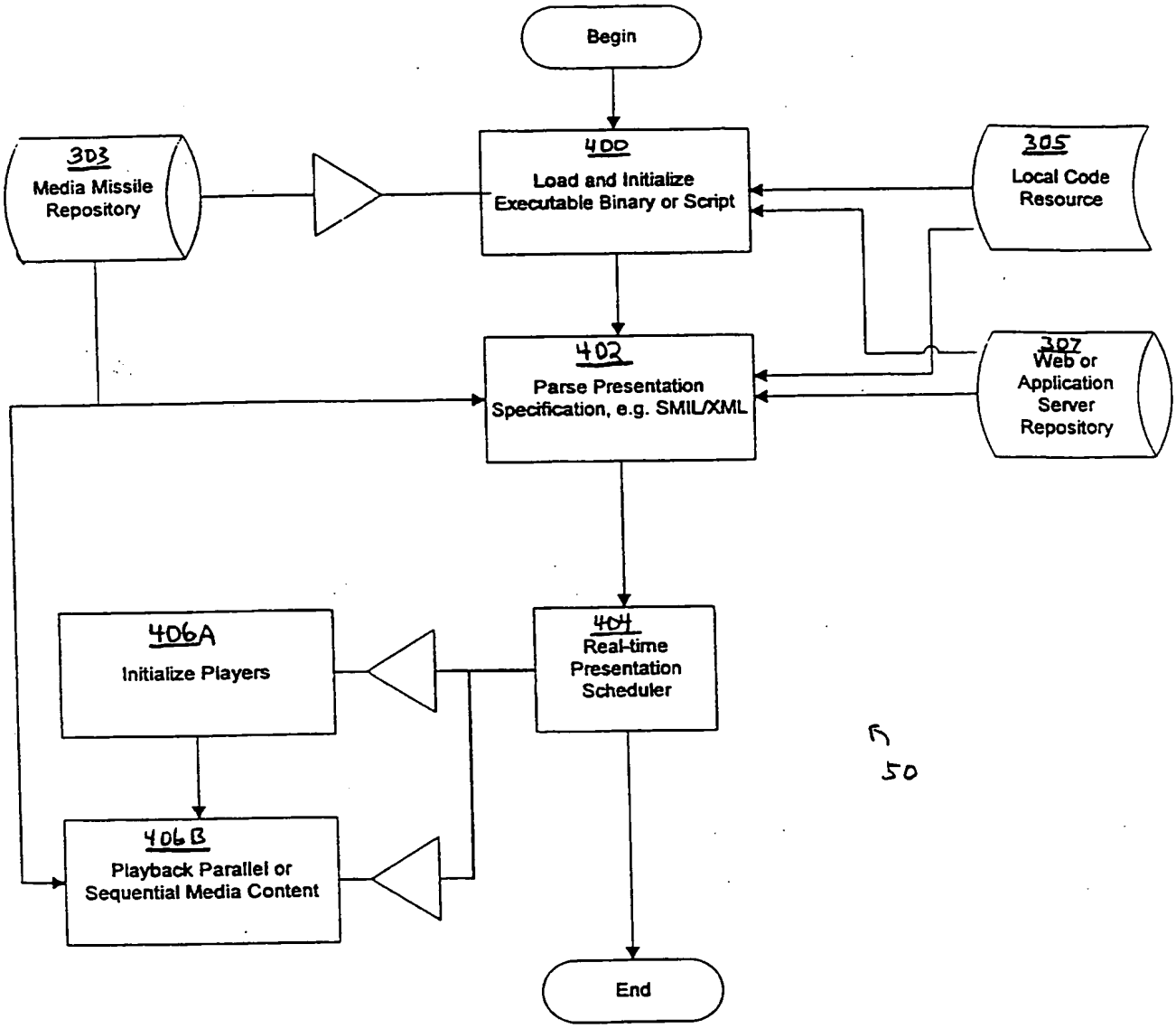

Begin
303
Media Missile Repository
400
Load and Initialize Executable Binary or Script
305
Local Code Resource
402
Parse Presentation Specification, e.g. SMIL/XML
307
Web or Application Server Repository
406A
Initialize Players
404
Real-time Presentation Scheduler
406B
Playback Parallel or Sequential Media Content
End
50

FIG. 4

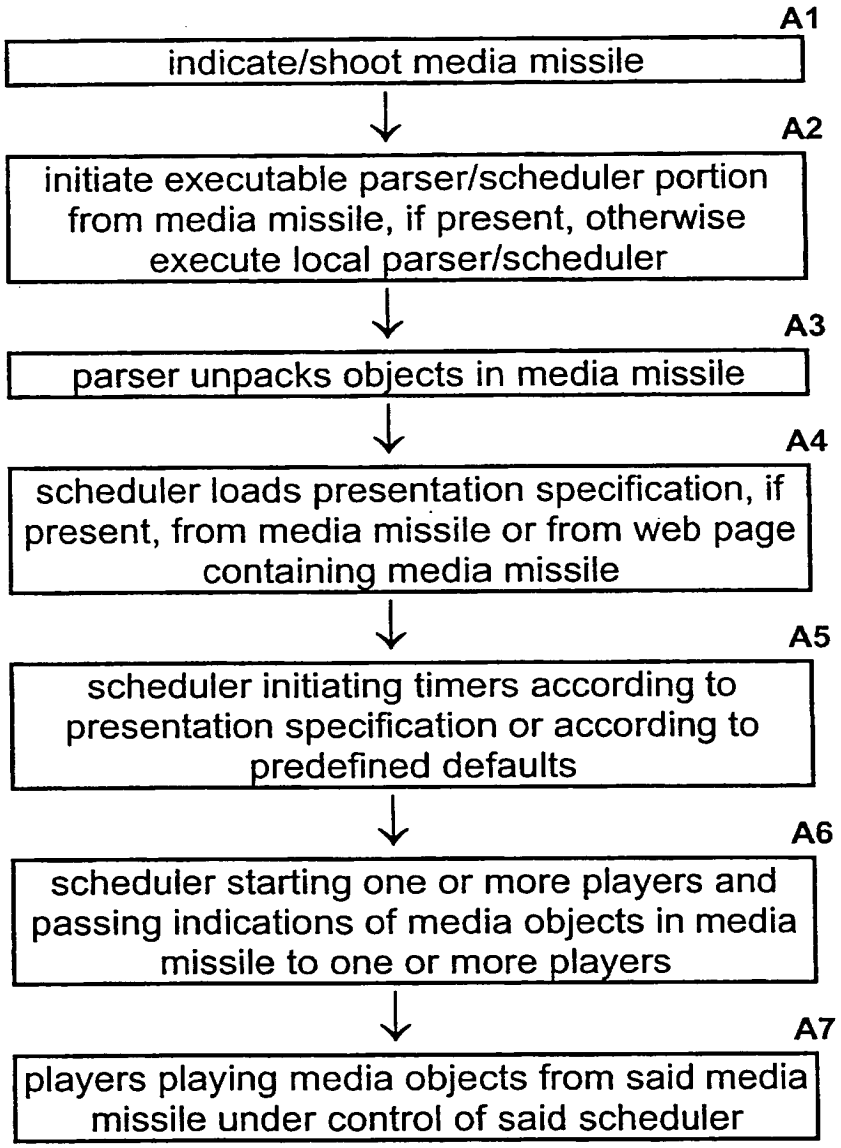
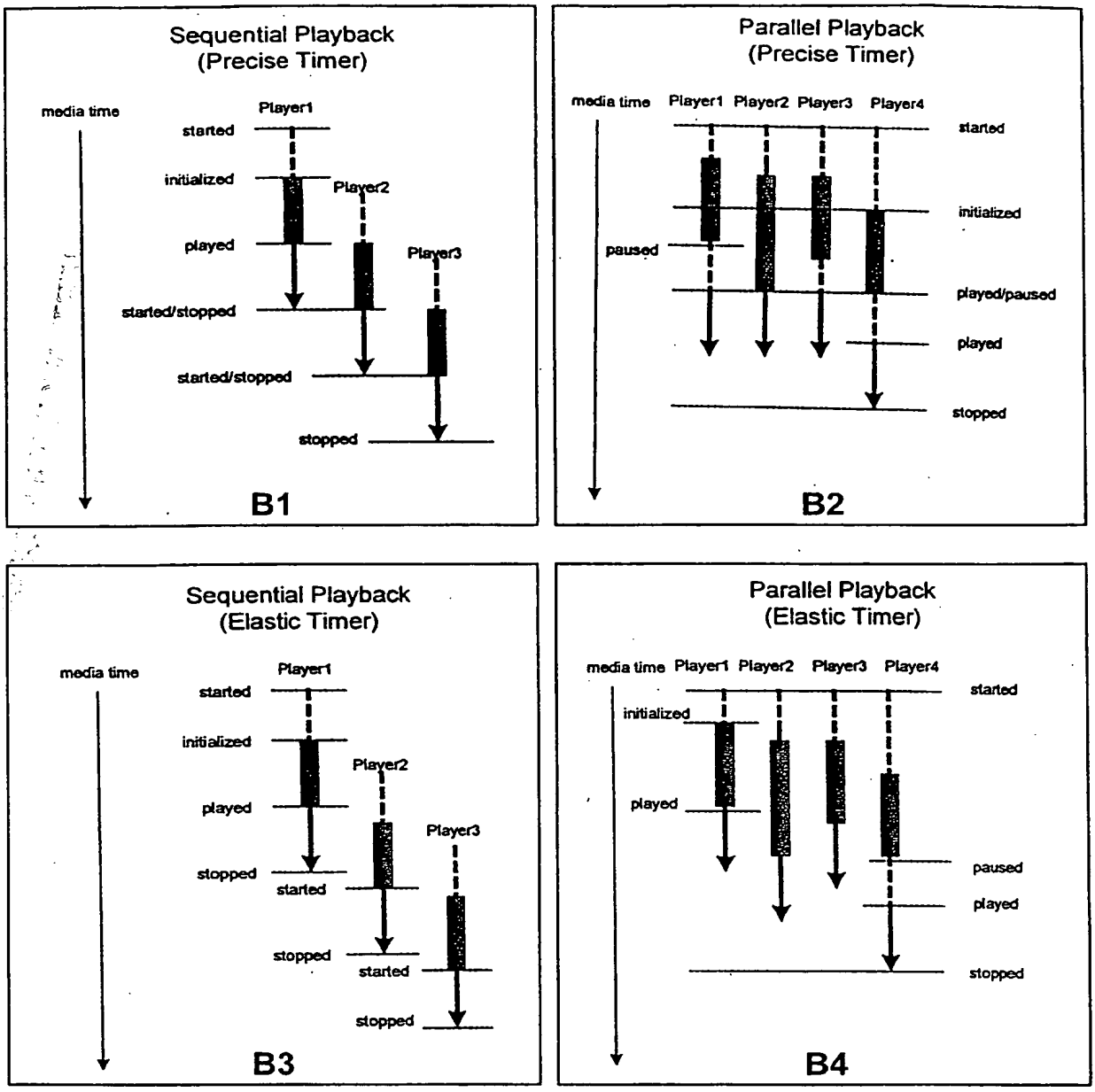


FIG. 5



Legend

initializing

Buffering

Playing

FIG. 6

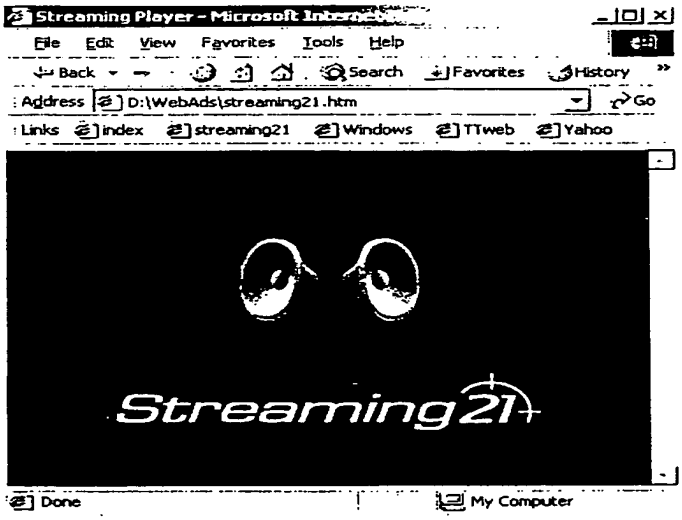


FIG. 7

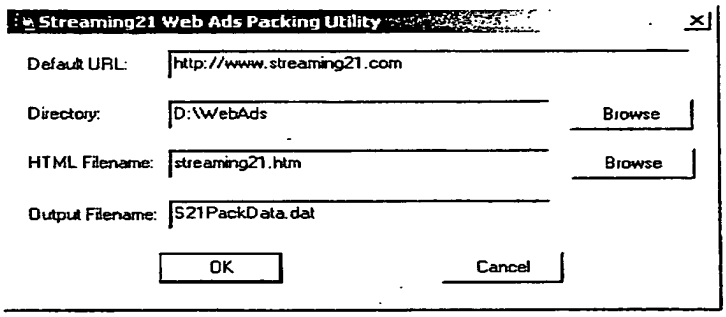


FIG. 8

Example METADATA block FORMAT

```
{ // metaDataBlock
  {  bit1   data_or_missile      // in this example always 0; => next
               block is data
     int63  metaDataBlkLen;      // sizeof(metaDataBlock);
     int128 signature;           // 0x03010101 => player version 3.01,
               // fileFormatId 1, version 1
     bit1   payload_or_url;      // always 1; => URL
     int63  defaultUrlLen;
     {  bit1   payload_or_url;   // always 0; => payload
        int63  filenameLen;
        bit1   payload_or_url;   // always 0; => payload
        int63  fileDataLen;
     }  fileNameDataLenStruct[numberOfFiles];          }
  // mediaDataBlock
  {  char defaultUrl[defaultUrlLen];
     {
        char filename[filenameLen];
        char fileContent[fileDataLen];
     }  fileNameDataStruct[numberOfFile];
  }  mediaDataBlock;
  // ending
  bit1  p_blk_exist;
  int63 eLength;                    }
// be aware the byte order difference in file vs in memory
// e.g. int value = READ(); File Hex Data "12 34 56 78";
//      value in Hex is 0x78563412;
```

FIG. 9

EXAMPLE METADATA BLOCK CONTENTS

Assumed values

```
Default URL is "http://www.abcCo.com"     strlen 20
c:\webAds\s21WebAds.htm            nameLen 13, size  2562 bytes     // the
              main page
c:\webAds\otherLinks.htm           nameLen 14, size  1548 bytes
c:\webAds\logo.gif                 nameLen  8, size  5874 bytes
c:\webAds\sponsor\abcCo.jpg        nameLen 17, size 18745 bytes
```

Metadata Media Missile values

```
bit1   data_or_missile            //     0
int63  metaDataBlkLen;            // (64/8)*(4+4*2) = 96 bytes
int128 signature;                 // 0x03010101
bit1   payload_or_url;            //     1
int63  defaultUrlLen;             //    20
bit1   payload_or_url;            //     0
int63  filenameLen1;              //    13
bit1   payload_or_url;            //     0
int63  fileDataLen1;              //  2562
bit1   payload_or_url;            //     0
int63  filenameLen2;              //    14
bit1   payload_or_url;            //     0
int63  fileDataLen2;              //  1548
bit1   payload_or_url;            //     0
int63  filenameLen3;              //     8
bit1   payload_or_url;            //     0
int63  fileDataLen3;              //  5874
bit1   payload_or_url;            //     0
int63  filenameLen4;              //    17
bit1   payload_or_url;            //     0
int63  fileDataLen4;              // 18745
// At 96 bytes; this is the media data
char  defaultUrl[20]              // "http://www.abcCo.com"
char  filename1[13]               // "s21WebAds.htm"    (simple
              encryption)
char  fileData1[2562]             // s21WebAds.htm's data
char  filename2[14]               // "otherLinks.htm"   (simple
              encryption)
char  fileData2[2562]             // otherLinks.htm's data
char  filename3[8]                // "logo.gif"
char  fileData3[2562]             // logo.gif's data    (simple
              encryption)
char  filename4[17]               // "sponsor\abcCo.jpg"
char  fileData4[2562]             //   sponsor\abcCo.jpg's   data   (simple
              encryption)
bit1  contination_bit             // 0
int63 eLength // 0
```

FIG. 10

MEDIA MISSILE BLOCK CONTENTS EXAMPLE

```
char    executablecode[nnnnnn];  // Executable code
bit1    P_spec_contination_bit;  // 1
int63   pLength;                    // pppp
char    presentation_spec[pppp]; // Presentation Specification
bit1    data_or_missile          //     0
int63   metaDataBlkLen;          // (64/8)*(4+4*2) = 96 bytes
int128  signature;                  // 0x03010101
bit1    payload_or_url;          //     1
int63   defaultUrlLen;           //    20
bit1    payload_or_url;          //     0
int63   filenameLen1;            //    13
bit1    payload_or_url;          //     0
int63   fileDataLen1;            //  2562
bit1    payload_or_url;          //     0
int63   filenameLen2;            //    14
bit1    payload_or_url;          //     0
int63   fileDataLen2;            //  1548
bit1    payload_or_url;          //     0
int63   filenameLen3;            //     8
bit1    payload_or_url;          //     0
int63   fileDataLen3;            //  5874
bit1    payload_or_url;          //     0
int63   filenameLen4;            //    17
bit1    payload_or_url;          //     0
int63   fileDataLen4;            // 18745
//
//
// At eLength + pLength plus 104  bytes; this is the media data; or note
            that
// if the data_or_missile bit = 1, this portion could begin another
// media missile.  In that case, the words after the signature block
            above will
// have the form:
//      bit1   payload_or_url;   //     0
//      int63  MediaMissileLen1; //  mmmm
// and the media data will include the object
//      char  mediamissile[mmmm]
//
//
char  defaultUrl[20]             // "http://www.abcCo.com"
char  filename1[13]              // "s21WebAds.htm"    (simple
            encryption)
char  fileData1[2562]            // s21WebAds.htm's data
char  filename2[14]              // "otherLinks.htm"   (simple
            encryption)
char  fileData2[1548]            // otherLinks.htm's data
char  filename3[8]               // "logo.gif"
char  fileData3[5874]            // logo.gif's data    (simple
            encryption)
char  filename4[17]              // "sponsor\abcCo.jpg"
char  fileData4[18745]           //   sponsor\abcCo.jpg's   data   (simple
            encryption)
bit1  continuation_bit           // 0
int63 eLength // nnnnnn
```

FIG. 11

S21PackData.dat]

```
000000  B0 00 00 00 00 00 00 00  01 01 01 03 00 00 00 00
000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 80
000020  0F 00 00 00 00 00 00 80  0F 00 00 00 00 00 00 00
000030  00 03 00 00 00 00 00 00  0C 00 00 00 00 00 00 00
000040  03 49 00 00 00 00 00 00  08 00 00 00 00 00 00 00
000050  31 13 00 00 00 00 00 00  BF B8 BE A9 AD A1 A5 A2
000060  AB FE FD E2 A4 B8 A1 BF  B8 BE A9 AD A1 A5 A2 AB
000070  FE FD E2 A4 B8 A1 F0 A4  B8 A1 A0 F2 C1 C6 C1 C6
000080  F0 A4 A9 AD A8 F2 C1 C6  F0 B8 A5 B8 A0 A9 F2 9F
000090  B8 BE A9 AD A1 A5 A2 AB  EC 9C A0 AD B5 A9 BE F0
0000a0  E3 B8 A5 B8 A0 A9 F2 C1  C6 F0 80 85 82 87 EC 9E
0000b0  89 80 F1 9F B8 B5 A0 A9  9F A4 A9 A9 B8 EC 84 9E
0000c0  89 8A F1 EE BC AD AB A9  E2 AF BF BF EE EC 98 95
0000d0  9C 89 F1 EE B8 A9 B4 B8  E3 AF BF BF EE F2 C1 C6
0000e0  C1 C6 F0 E3 A4 A9 AD A8  F2 C1 C6 C1 C6 F0 AE A3
0000f0  A8 B5 EC AE AB AF A3 A0  A3 BE F1 EE EF FC FC FC
000100  FC FC FC EE EC B8 A9 B4  B8 F1 EE EF 8A 8A 8A 8A
000110  8A 8A EE EC A0 A5 A2 A7  F1 EE EF 8A 8A 8A 8A 8A
000120  8A EE EC BA A0 A5 A2 A7  F1 EE EF 8A 8A 8A 8A 8A
000130  8A EE EC AD A0 A5 A2 A7  F1 EE EF 8A 8A 8A 8A 8A
000140  8A EE EC 81 8D 9E 8B 85  82 9B 85 88 98 84 F1 EE
000150  FC EE EC 80 89 8A 98 81  8D 9E 8B 85 82 F1 EE FC
000160  EE EC 81 8D 9E 8B 85 82  84 89 85 8B 84 98 F1 EE
000170  FC EE EC 98 83 9C 81 8D  9E 8B 85 82 F1 EE FC EE
000180  F2 C1 C6 C1 C6 F0 A8 A5  BA EC AD A0 A5 AB A2 F1
000190  EE AF A9 A2 B8 A9 BE EE  F2 C1 C6 EC EC F0 AF A9
0001a0  A2 B8 A9 BE F2 C1 C6 EC  EC F0 B8 AD AE A0 A9 EC
0001b0  AE A3 BE A8 A9 BE F1 EE  FC EE EC AF A9 A0 A0 BC
0001c0  AD A8 A8 A5 A2 AB F1 EE  FC EE EC AF A9 A0 A0 BF
0001d0  BC AD AF A5 A2 AB F1 EE  FC EE EC BB A5 A8 B8 A4
0001e0  F1 EE FD FC FC E9 EE EC  A4 A9 A5 AB A4 B8 F1 EE
0001f0  FD FC FC E9 EE F2 C1 C6  EC EC EC EC F0 B8 BE F2
000200  C1 C6 EC EC EC EC EC EC  F0 B8 A8 F2 C1 C6 EC EC
000210  EC EC EC EC EC EC F0 BC  EC AD A0 A5 AB A2 F1 EE
000220  AF A9 A2 B8 A9 BE EE F2  F0 AD EC A4 BE A9 AA F1
000230  EE A4 B8 B8 BC F6 E3 E3  BB BB BB E2 BF B8 BE A9
000240  AD A1 A5 A2 AB FE FD E2  AF A3 A1 EE EC B8 AD BE
        ................
        ................
        ................
        ................
        ................
        ................
        ................
```

FIG. 12

FIG. 13

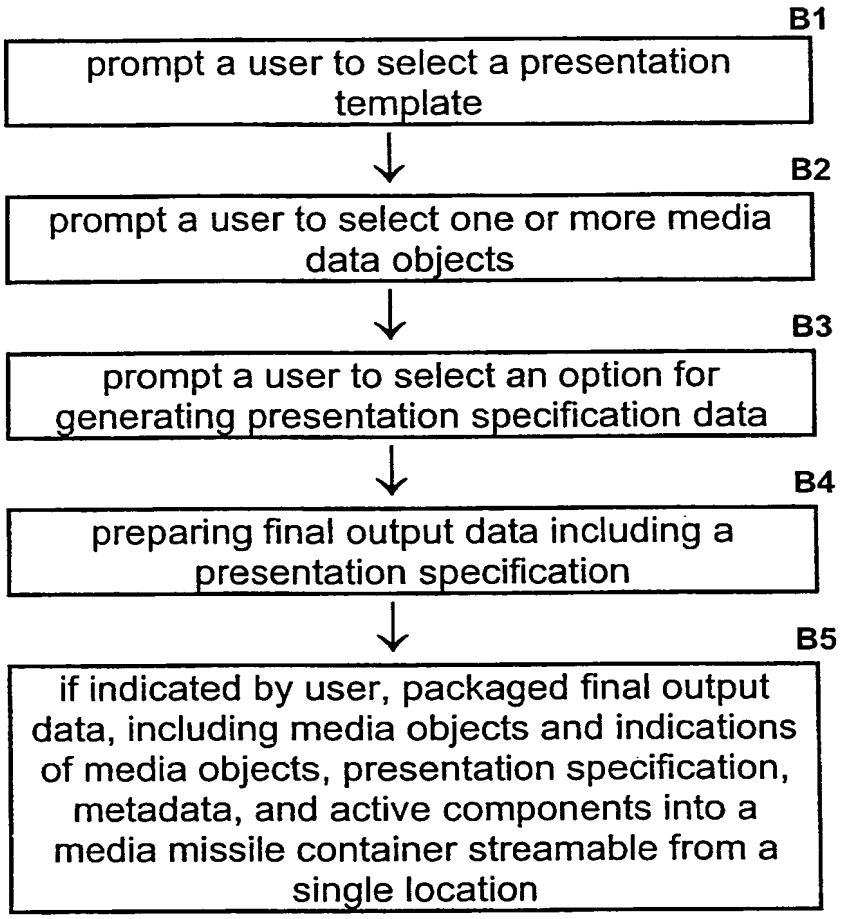

B1
prompt a user to select a presentation template
B2
prompt a user to select one or more media data objects
B3
prompt a user to select an option for generating presentation specification data
B4
preparing final output data including a presentation specification
B5
if indicated by user, packaged final output data, including media objects and indications of media objects, presentation specification, metadata, and active components into a media missile container streamable from a single location

FIG. 14

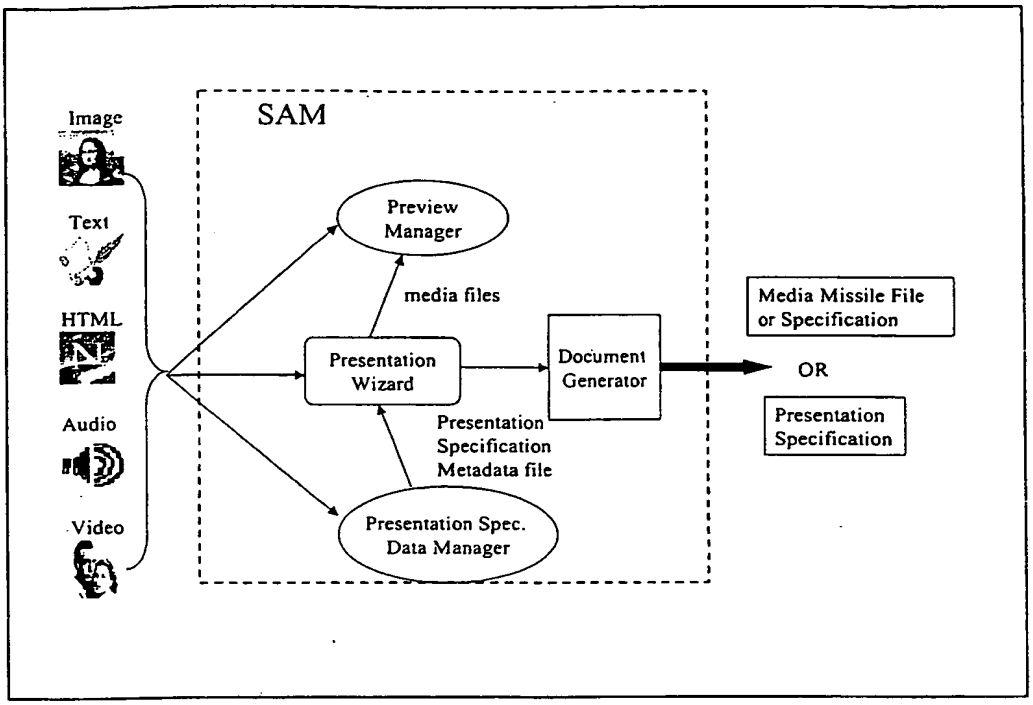


FIG. 15

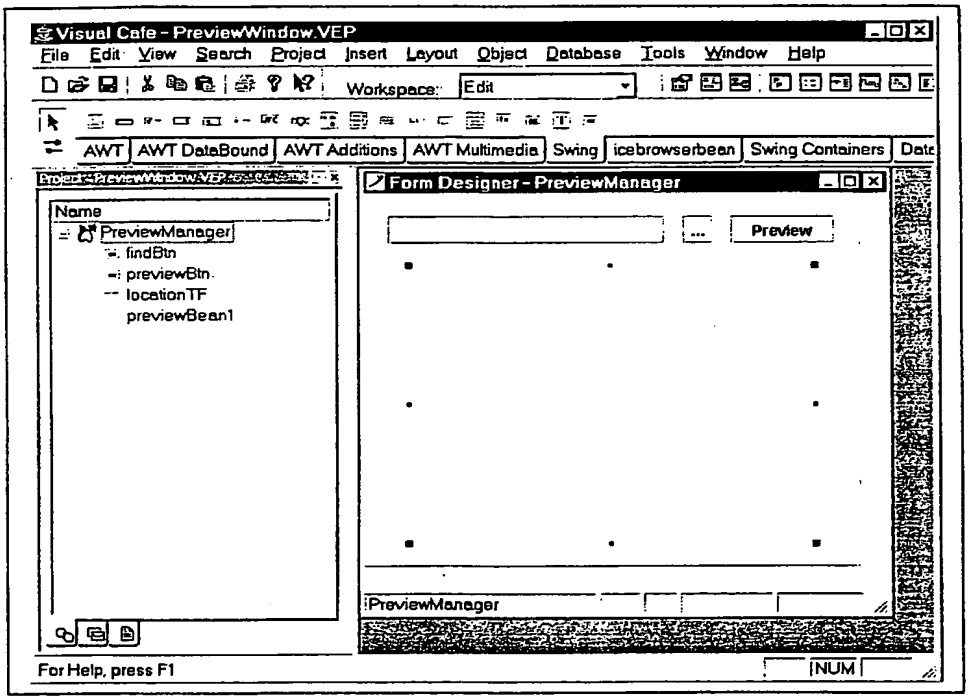


FIG. 16 Preview Manager GUI

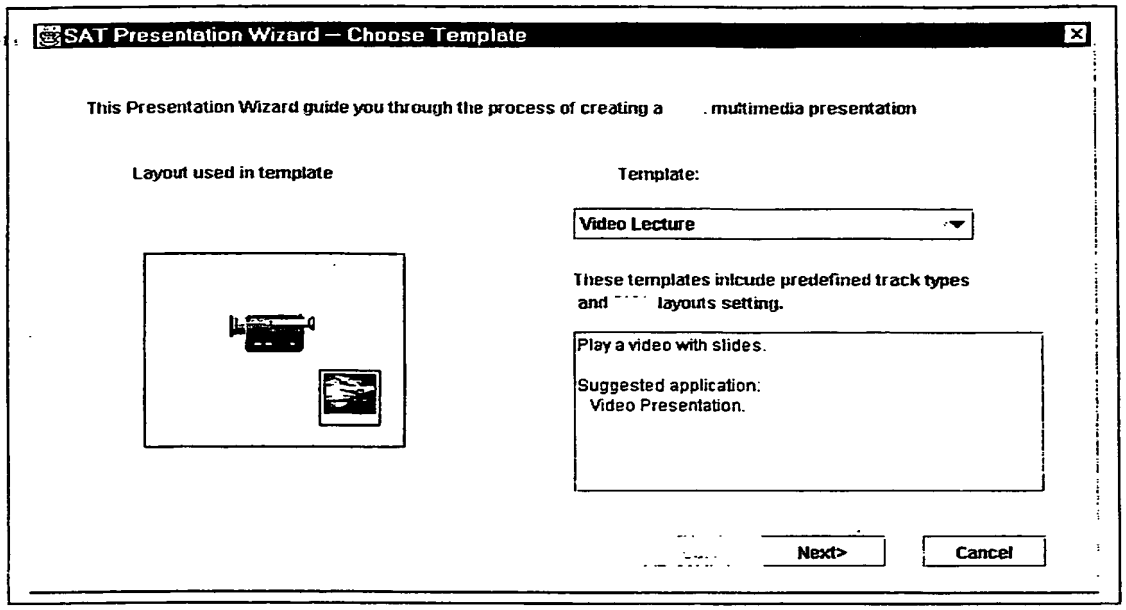


FIG. 17A Choosing a Video Lecture Template

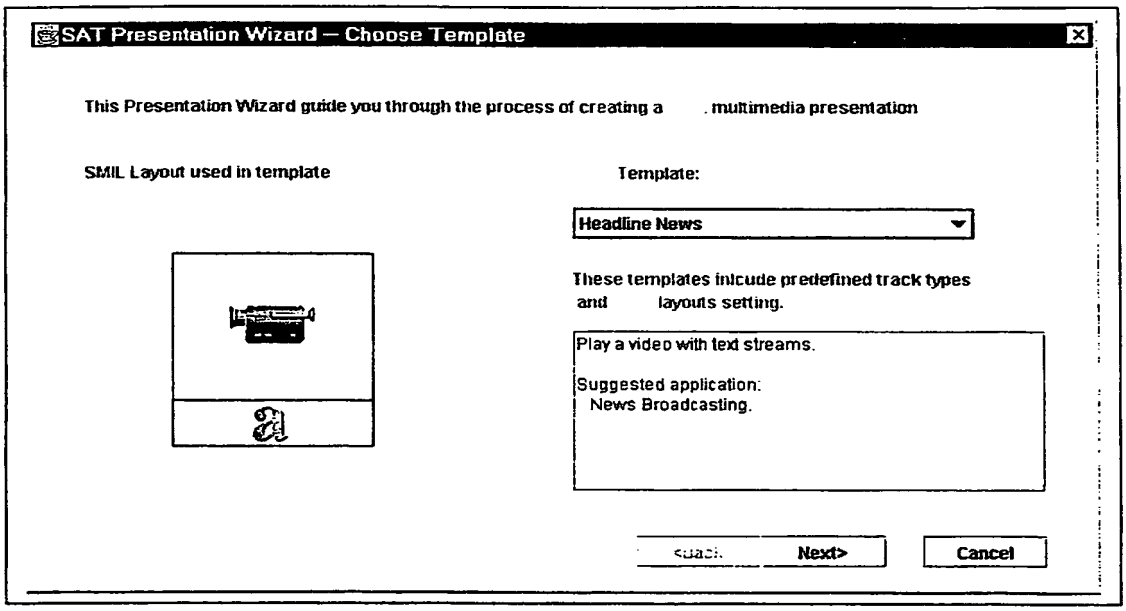


FIG. 17B Choosing a Headline News Template

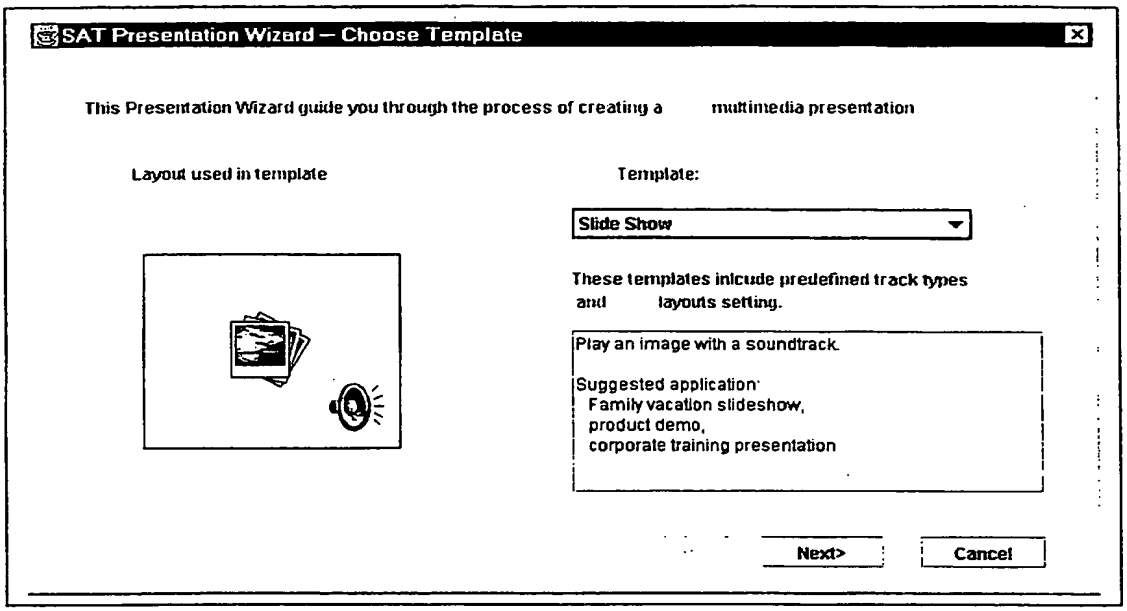


FIG. 17C Choosing a Slide Show Template

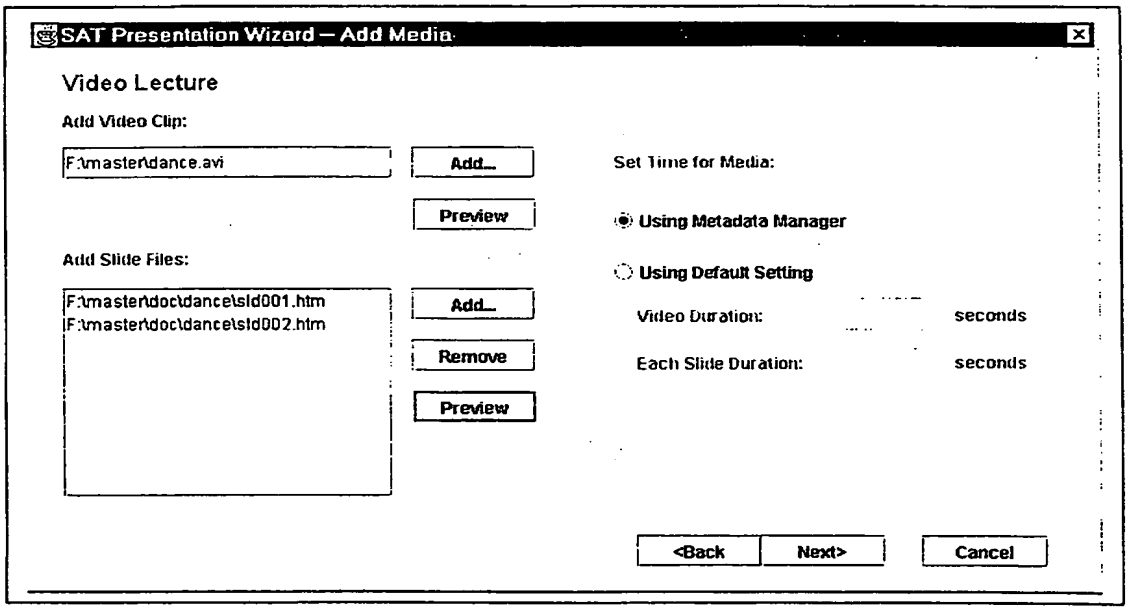


FIG. 18A Video Lecture Add And Preview Media

FIG. 18B Video Lecture Template Preview Selected Slides

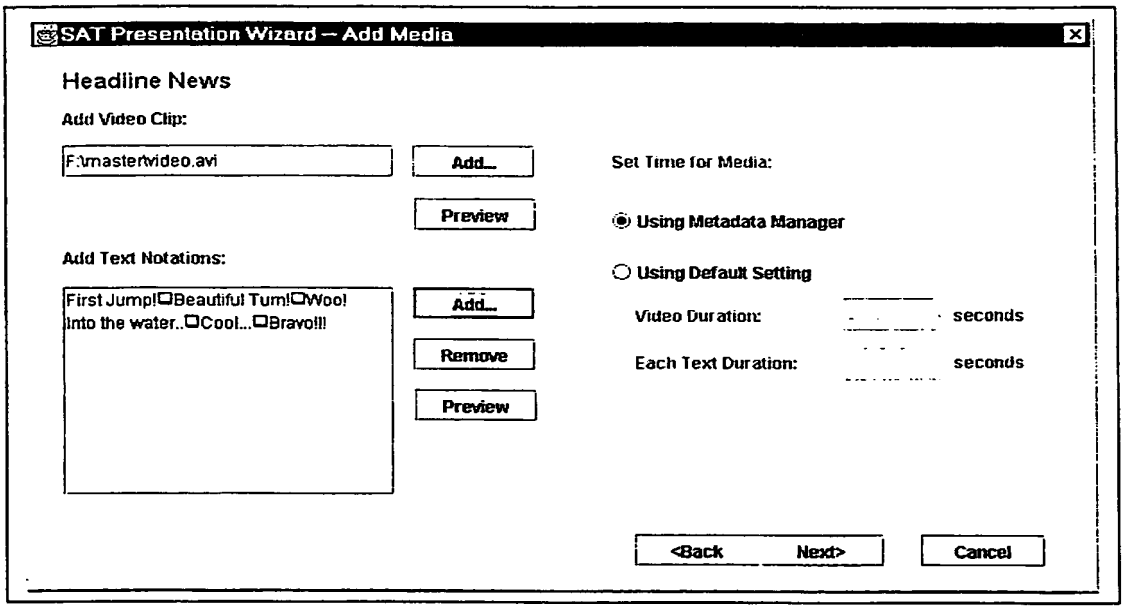


FIG. 19A Headline News Template – Add And Preview Media

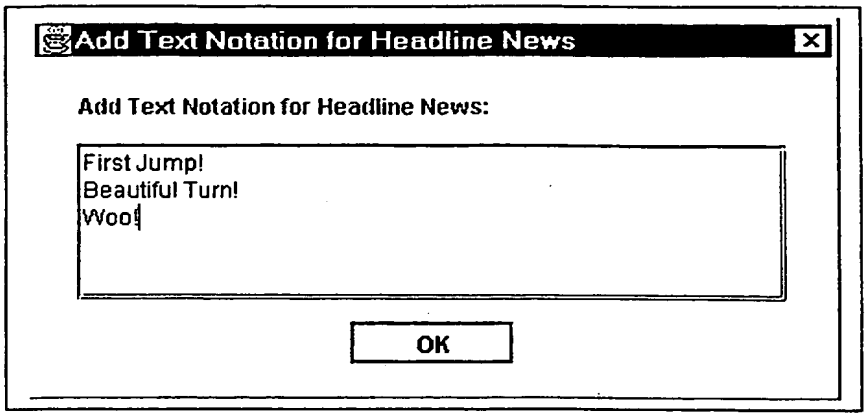


FIG. 19B Add Text Notation For Headline News

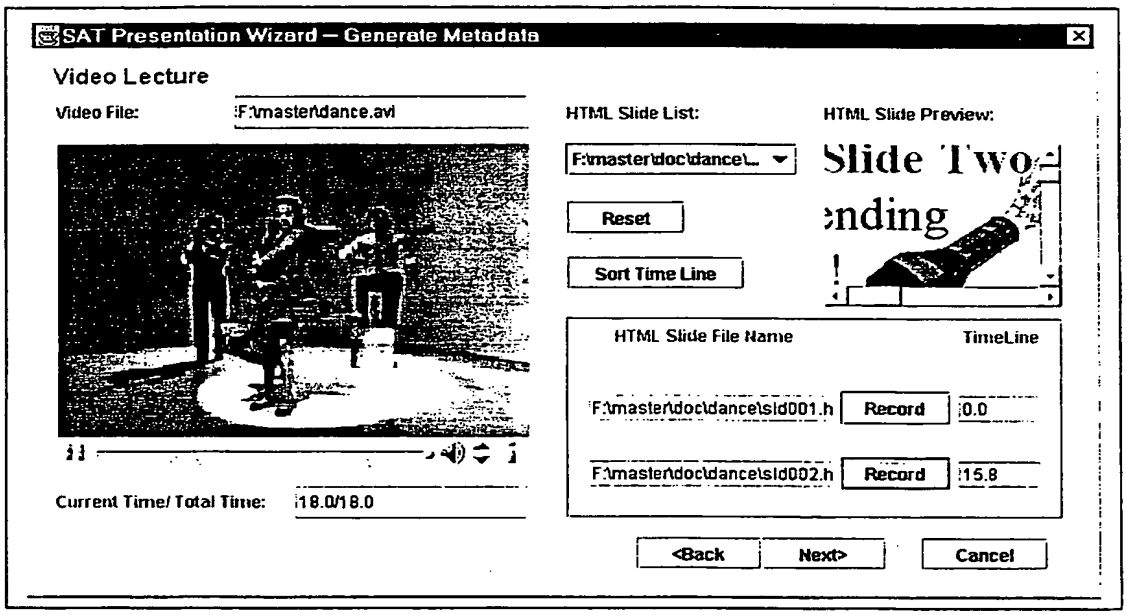


FIG. 20A Video Lecture Template – Create Presentation Specification Data

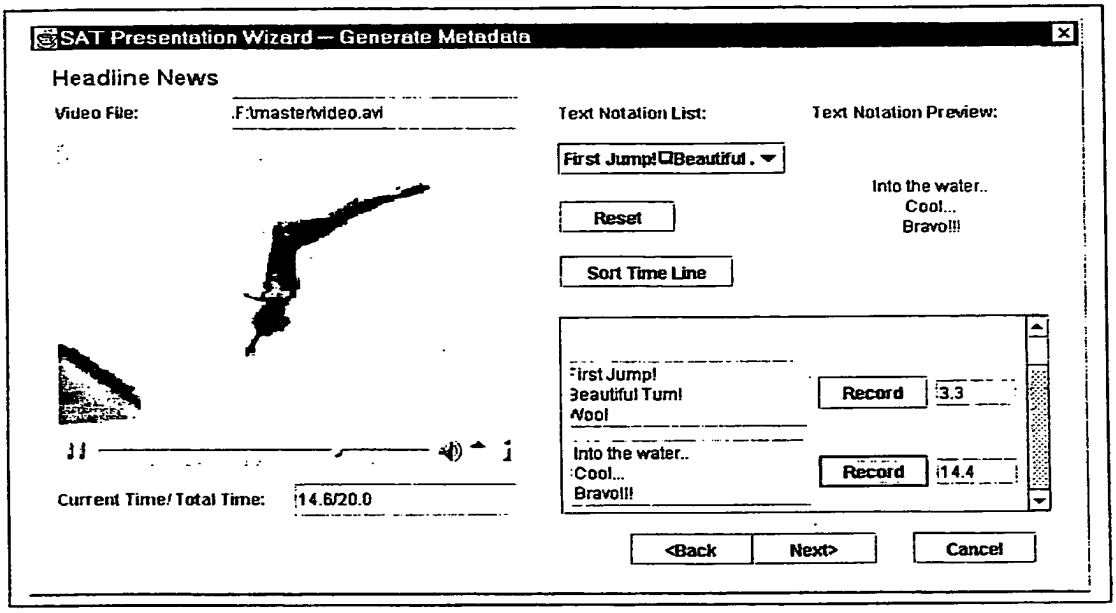


FIG. 20B Headline News Template – Create Presentation Specification Data

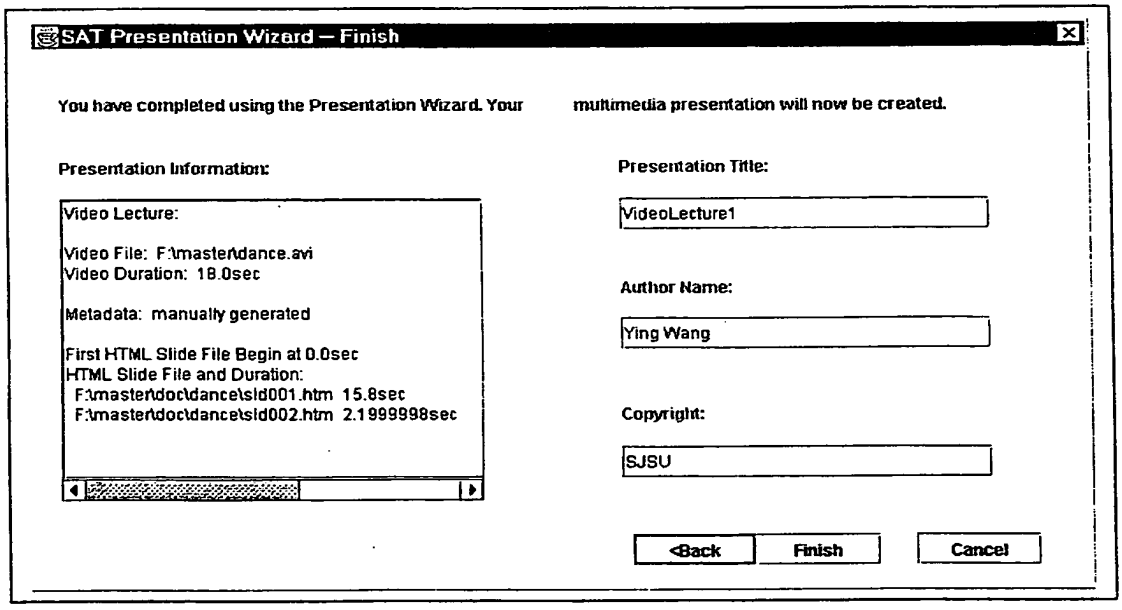


FIG. 21 Video Lecture Template - Generate Document

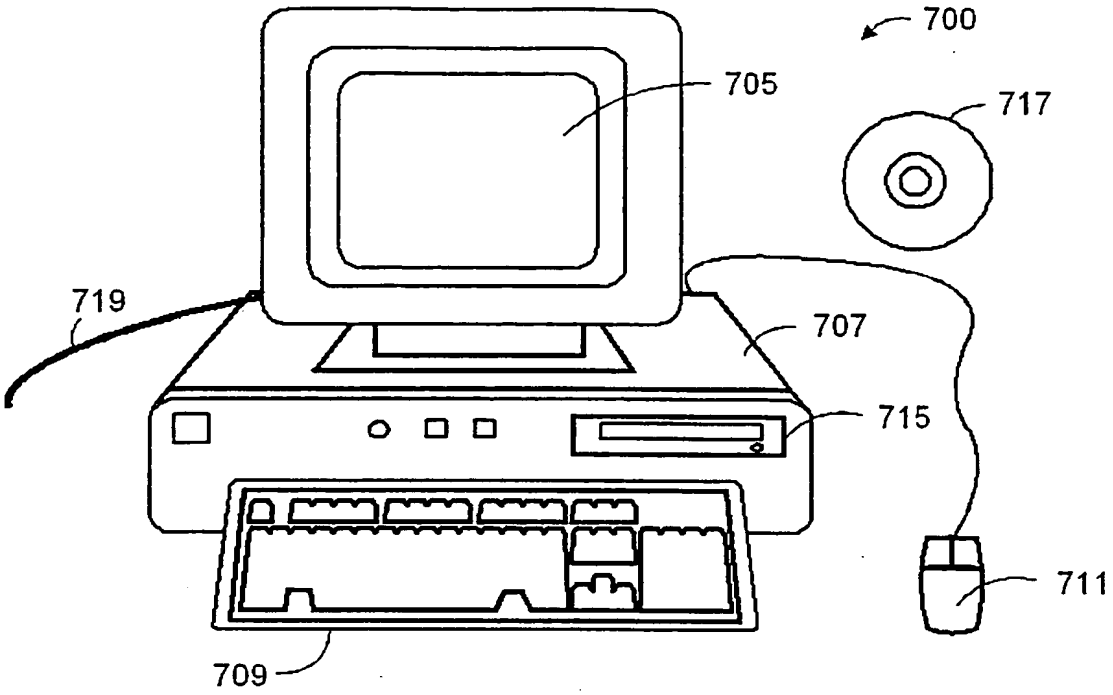


FIG. 22

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 00/27634

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04N7/24

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04N G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data, PAJ, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	MEYER-BOUDNIK T ET AL: "MHEG EXPLAINED" IEEE MULTIMEDIA,IEEE COMPUTER SOCIETY,US, vol. 2, no. 1, 21 March 1995 (1995-03-21), pages 26-38, XP000500084 ISSN: 1070-986X	1,6-13, 15, 17-34, 39,40, 47,48, 50,51, 60,61, 63, 65-68, 72,73, 75-77, 79-82
Y	the whole document	3-5,16, 35-38, 49,78
A		2,14, 41-46, 52-59, 62,64,

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
28 February 2001	13/03/2001

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	La, V

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 00/27634

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
		69-71, 74,83

Y	"Seiko Epson and Spyglass collaborate on new wireless communications and computing device, New handheld innovation will use Spyglass Prism to deliver custom content services to Japanese consumers" OPENTV NEWS & EVENTS, PRESS ARCHIVES, 'Online! 23 August 1999 (1999-08-23), XP002161629 Retrieved from the Internet: <URL:www.opentv.com> the whole document	3-5

Y	WO 99 22563 A (KONINKL PHILIPS ELECTRONICS NV ;PHILIPS SVENSKA AB (SE)) 14 May 1999 (1999-05-14) page 1, line 1 -page 4, line 31 page 14, line 15 - line 17	16,49,78
A	abstract	1-15, 17-48, 50-77, 79-83

Y	JUE XIA ET AL: "Design and implementation of a SMIL player" COLOR IMAGING: DEVICE-INDEPENDENT COLOR, COLOR HARDCOPY, AND GRAPHIC ARTS IV, SAN JOSE, CA, USA, 26-29 JAN. 1999, vol. 3648, pages 382-389, XP000986870 Proceedings of the SPIE - The International Society for Optical Engineering, 1998, SPIE-Int. Soc. Opt. Eng, USA ISSN: 0277-786X section 1 "Introduction"	35-38,45
A	section 4. "Design and implementation"	1-34, 39-44, 46-83

	-/--	

1

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 00/27634

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
X	COLAITIS F ET AL: "MHEG AND ITS PROFILE FOR ITV APPLICATIONS" IEE COLLOQUIUM ON INTERACTIVE TELEVISION, 2 October 1995 (1995-10-02), XP000646001	1,2, 6-13,15, 17-34, 39-42, 44, 46-48, 50,51, 60,61, 63, 65-68, 72,73, 75-77, 79-82
Y	the whole document	45
A		3-5,14, 16, 35-38, 43,49, 52-59, 62,64, 69-71, 74,78,83

X	"OpenAuthor, Overview, Technical White Paper" OPENTV, INC, July 1998 (1998-07), XP002126675	1,6-13, 15, 17-34, 39,40, 47,48, 50,51, 60,61, 63,67, 68,72, 73, 75-77, 79-82
A	section 1 "What is OpenAuthor ?" section 3 "Creating applications with OpenAuthor"	2-5,14, 16, 35-38, 41-46, 49, 52-59, 62, 64-66, 69-71, 74,78,83

	-/--	

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 00/27634

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 768 539 A (DESA COLIN JOSEPH ET AL) 16 June 1998 (1998-06-16) column 5, line 8 -column 6, line 55 column 9, line 57 -column 10, line 28 abstract	1,7-9, 11-13, 15, 17-25, 27-30, 32-34, 47,48, 50,51

X	KERAMANE C ET AL: "Operator based composition of structured multimedia presentations" FROM MULTIMEDIA SERVICES TO NETWORK SERVICES. 4TH INTERNATIONAL COST 237 WORKSHOP. PROCEEDINGS, FROM MULTIMEDIA SERVICES TO NETWORK SERVICES. 4TH INTERNATIONAL COST 237 WORKSHOP. PROCEEDINGS, LISBOA, PORTUGAL, 15-19 DEC. 1997, pages 1-17, XP000986657 1997, Berlin, Germany, Springer-Verlag, Germany ISBN: 3-540-63935-7 section I "Introduction"	1,18,30, 47,48
A	abstract	2-17, 19-29, 31-46, 49-83

X	JOURDAN M ET AL: "Authoring SMIL documents by direct manipulations during presentation" WORLD WIDE WEB, 1999, BALTZER, NETHERLANDS, vol. 2, no. 4, pages 179-190, XP000986864 ISSN: 1386-145X	60,62, 63, 65-69, 72-74, 80-83
A	section 3 "SMIL : what is it and how do we edit it ?"	1-59,61, 64,70, 71,75-79

	-/--	

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 00/27634

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P,X	SHIM S S Y ET AL: "Template based Synchronized Multimedia Integration Language authoring tool" INTERNET IMAGING, SAN JOSE, CA, USA, 26-28 JAN. 2000, vol. 3964, pages 134-142, XP000986660 Proceedings of the SPIE - The International Society for Optical Engineering, 2000, SPIE-Int. Soc. Opt. Eng, USA ISSN: 0277-786X section 1 "Introduction" section 4 "Design and implementation" figures 1,2 ---	60,62, 63, 65-68, 72-74, 80-83
A	US 5 586 235 A (KAUFFMAN IVAN J) 17 December 1996 (1996-12-17) section "Summary of the invention" column 5, line 65 -column 6, line 36 abstract ---	1-83
A	WO 96 42144 A (NOKIA OY AB ;SALOMAEKI ARI (FI)) 27 December 1996 (1996-12-27) the whole document ---	1-83
A	US 5 881 244 A (IWAMOTO AKIRA ET AL) 9 March 1999 (1999-03-09) abstract ---	1-83
A	WO 98 29835 A (STARNET INC) 9 July 1998 (1998-07-09) abstract ---	1-83
A	HOUNAM D: "Presentation panache (PowerPoint software package)" WHICH COMPUTER?, NOV. 1988, UK, page 23, 26, 28 XP000986647 ISSN: 0140-3435 middle column, paragraph 3 ---	60
A	BLAKOWSKI G ET AL: "TOOL SUPPORT FOR THE SYNCHRONIZATION AND PRESENTATION OF DISTRIBUTED MULTIMEDIA" COMPUTER COMMUNICATIONS,NL,ELSEVIER SCIENCE PUBLISHERS BV, AMSTERDAM, vol. 15, no. 10, 1 December 1992 (1992-12-01), pages 611-618, XP000321682 ISSN: 0140-3664 the whole document ---	1-83

-/--

1

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 00/27634

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	HOFRICHTER K: "MHEG 5 - STANDARDIZED PRESENTATION OBJECTS FOR THE SET TOP UNIT ENVIRONMENT" INTERACTIVE DISTRIBUTED MULTIMEDIA SYSTEMS AND SERVICES,XX,XX, 4 March 1996 (1996-03-04), pages 33-44, XP000672134 section 4.2 "The MHEG engine" ---	1-83
A	TEN KATE W ET AL: "Presenting multimedia on the Web and in TV broadcast" MULTIMEDIA APPLICATIONS, SERVICES AND TECHNIQUES - ECMAST'98. THIRD EUROPEAN CONFERENCE. PROCEEDINGS, MULTIMEDIA APPLICATIONS, SERVICES AND TECHNIQUES - ECMAST '98 THIRD EUROPEAN CONFERENCE PROCEEDINGS, BERLIN, GERMANY, 26-28 MAY 1998, pages 56-69, XP000984176 1998, Berlin, Germany, Springer-Verlag, Germany ISBN: 3-540-64594-2 section 2 "MHEG-5" section 3 "SMIL" section 4 "Translating SMIL into MHEG-5" ---	1-83
A	ROISIN C: "Authoring structured multimedia documents" SOFSEM '98: THEORY AND PRACTICE OF INFORMATICS. 25TH CONFERENCE ON CURRENT TRENDS IN THEORY AND PRACTICE OF INFORMATICS. PROCEEDINGS, SOFSEM '98: THEORY AND PRACTICE OF INFORMATICS. 25TH CONFERENCE ON CURRENT TRENDS IN THEORY AND PRACTICE OF INFORMAT, pages 222-239, XP000987024 1998, Berlin, Germany, Springer-Verlag, Germany ISBN: 3-540-65260-4 section 4.1 "Multimedia authoring requirements" -----	60-83

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 00/27634

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9922563 A	14-05-1999	CN 1248336 T	22-03-2000
		EP 0974110 A	26-01-2000
US 5768539 A	16-06-1998	US 5666293 A	09-09-1997
		US 5734589 A	31-03-1998
		US 5635979 A	03-06-1997
		US 5978855 A	02-11-1999
		AU 2657995 A	21-12-1995
		WO 9533338 A	07-12-1995
US 5586235 A	17-12-1996	NONE	
WO 9642144 A	27-12-1996	FI 98175 B	15-01-1997
		AU 6127696 A	09-01-1997
		EP 0872053 A	21-10-1998
		JP 11510971 T	21-09-1999
		US 6094661 A	25-07-2000
US 5881244 A	09-03-1999	JP 9074556 A	18-03-1997
		JP 9081481 A	28-03-1997
		JP 9154131 A	10-06-1997
		EP 0762777 A	12-03-1997
		US 5802315 A	01-09-1998
WO 9829835 A	09-07-1998	AU 4003595 A	31-07-1998

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

CORRECTED VERSION

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 April 2001 (12.04.2001)

PCT

(10) International Publication Number
WO 01/26378 A1

(51) International Patent Classification[7]: H04N 7/24

(21) International Application Number: PCT/US00/27634

(22) International Filing Date: 5 October 2000 (05.10.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/157,998	6 October 1999 (06.10.1999)	US
09/679,763	5 October 2000 (05.10.2000)	US

(71) Applicant: STREAMING21, INC. [US/US]; Suite 1, 170 Knowles Drive, Los Gatos, CA 95032 (US).

(72) Inventors: LEE, Yen-Jen; 3643 Madison Common, Fremont, CA 94538 (US). SHIM, Sang, Yup; 1157 Pheasant Hill Drive, San Jose, CA 95120 (US).

(74) Agent: HSUE, James, S.; Skjerven Morrill MacPherson LLP, Suite 700, 25 Metro Drive, San Jose, CA 95110 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

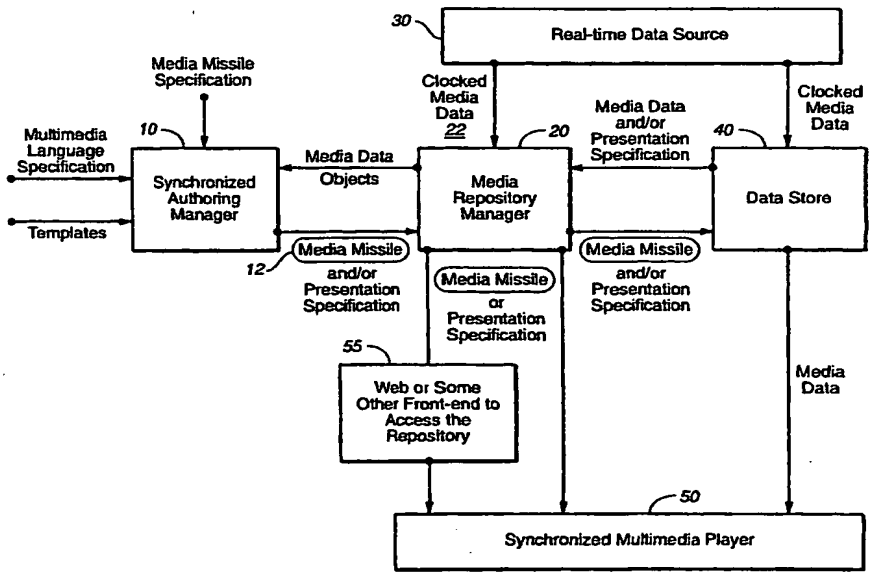
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(48) Date of publication of this corrected version:
27 September 2001

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR MANAGING STREAMING DATA



(57) Abstract: A streaming media structure methods and systems for creating, managing, and delivering streaming media is disclosed. The streaming media structure is provided to create, deliver, reassemble, and render synchronized multimedia content with needed machine-aware components through streaming services.

WO 01/26378 A1

(15) **Information about Correction:**
see PCT Gazette No. 39/2001 of 27 September 2001, Section II

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

VOL. E86-B NO. 6 JUNE 2003

IEICE Transactions on Communications

Original Contributions in English

The IEICE Transactions on Communications has been selected for coverage in the following ISI (Institute for Scientific Information) products/services.

- Science Citation Index*
- Science Citation Index Expanded (included in "Web of Science")
- Current Contents (Edition : Engineering, Computing and Technology)
- ISI Alerting Services

*Science Citation Index is a subset of SCIE, covering 3500 of the leading scientific and technical journals. Inclusion in this particular database is very competitive.

Publication of this journal is partly supported by the Grant-in-Aid for Publication of Scientific Research Result from the Japan Society for the Promotion of Science.

CONTENTS

PAPER Special Issue on Content Delivery Networks

Content Delivery Network Architecture for Mobile Streaming Service Enabled by SMIL Modification

Takeshi YOSHIMURA[†a)], *Regular Member*, Yoshifumi YONEMOTO[†], *Nonmember*, Tomoyuki OHYA[†], Minoru ETOH[†], *Regular Members*, *and* Susie WEE[††], *Nonmember*

SUMMARY In this paper, we present a CDN (Content Delivery Network) architecture for mobile streaming service in which content segmentation, request routing, pre-fetch scheduling, and session handoff are controlled by SMIL (Synchronized Multimedia Integration Language) modification. In this architecture, mobile clients simply follow modified SMIL files downloaded from a portal server; these modifications enable multimedia content to be delivered to the mobile clients from the *best* surrogates in the CDN. The key components of this architecture are 1) content segmentation with SMIL modification, 2) on-demand rewriting of URLs in SMIL, 3) pre-fetch scheduling based on timing information derived from SMIL, and 4) SMIL updates by SOAP (Simple Object Access Protocol) messaging for session handoffs due to client mobility. This architecture enhances streaming media quality for mobile clients while utilizing network resources efficiently and supporting client mobility in an integrated and practical way. The current status of our prototype on a mobile QoS testbed "MOBIQ" is also reported in this paper.

key words: CDN, mobile network, streaming media, SMIL

1. Introduction

With the birth and growth of the Internet and high-speed access links, Internet users can enjoy large amounts of web content on the Internet. However, network congestion and server overload have become major concerns for content delivery. To mitigate these effects, CDNs (Content Delivery Networks) are attracting a great deal of attention. In CDNs, web content is distributed to cache servers located at the edge of the Internet close to clients, and the cache servers deliver content to clients if they cached the requested content beforehand. Since the cached content does not have to be delivered from the origin servers to clients, this leads to lower latency for users, better network resource utilization for network operators, and scalable service provisioning for content providers. At the moment, several commercial companies provide CDN service on worldwide scales [1].

In mobile environments, Internet access has become very popular as represented by i-mode [2] and WAP (wireless access protocol) [3]. In addition, IMT-2000 (International Mobile Telecommunications-2000) has begun in Japan [4], and users can now access the Internet at up to 384 kbit/s through mobile devices. In the near future, mobile CDNs will be needed with the increasing number of IMT-2000 users.

At the same time, multimedia streaming service is becoming popular over the Internet. Mobile networks are also expected to provide IP-based multimedia streaming service in addition to web access service. From this requirement, 3GPP (3rd Generation Partnership Project), a standardization body of IMT-2000, has defined "packet streaming service (PSS) specifications [5]" as a standard for streaming service over 3G mobile networks.

Streaming service, however, presents a lot of challenges for network engineers. Unlike TCP applications, streaming service requires a certain amount of bandwidth to ensure the bit-rate needed by each media stream and the strict delay variation (i.e., jitter) needed to avoid buffer underflow at streaming clients. For such streaming service, CDNs are a very effective solution to relieve network congestion and to keep jitter at tolerable levels. Several papers have already considered how to cache streaming content effectively. In [6] [8], streaming content is encoded with layered coding techniques and a proxy cache manages the number of layers to be stored for each stream. [9] proposed a prefix caching scheme that stores only the beginning of the content to minimize storage while decreasing waiting time for playback. [10] also utilizes the idea of prefix caching and describes resource management issues on cache systems. Selective caching was proposed in [11] as a generalization of prefix caching by storing various segments of the stream, specifically those that are subject to causing buffer underflow at clients. In [12], streaming content is divided into variable-sized segments in order to improve caching efficiency. Content-aware segmentation and a cache system using video summarization were proposed in [13]. In this system, streaming content is segmented at key frames and users can start streaming from any segment while watching the key frames.

Although these segment caching technologies have potential for enhancing network resource utilization and caching efficiency, they do not describe how to manage the segmented content and how to specify the

Manuscript received September 9, 2002.
Manuscript revised January 18, 2003.

[†]The authors are with Multimedia Laboratories, NTT DoCoMo, Inc., Yokosuka-shi, 239-8536 Japan.
[††]The author is with Hewlett-Packard Laboratories, Palo Alto, CA, 94304-1126 USA.
a) E-mail: yoshi@spg.yrp.nttdocomo.co.jp

locations of the proxy caches. Especially in mobile networks, since network and radio link conditions are dynamically changed depending on the time and place, it is required to specify the best cache locations flexibly. In addition, the best cache locations could be changed during a streaming session when a mobile client moves from one location to another location. The mechanism to update the cache locations is necessary for a mobile streaming media CDN. In [14], a streaming CDN architecture that has content naming, content management, and redirection mechanisms is described, but it does not consider network dynamics and client mobility expected in mobile networks.

The main goal of our work is to design a mobile streaming media CDN (MSM-CDN) that enhances streaming media quality in environments including mobile, while utilizing network resources efficiently and supporting client mobility in an integrated and practical way. To achieve this, we propose a dynamic SMIL framework in which all of the CDN technologies including content segmentation, request routing, pre-fetch control, and session handoff, are controlled by SMIL (Synchronized Multimedia Integration Language) [15] modification. The reasons for focusing on SMIL are itemized below.

- Since SMIL includes content location information, client requests can flexibly be redirected to the best cache servers by replacing the original content locations by the cache locations selected based on the network and caching conditions.
- To improve caching efficiency, large streaming content is likely to be divided into several segments. In this case, SMIL can easily manage the timing and spatial relations among these segments.
- SMIL makes user access patterns more predictable because it provides timing information about when a client is likely to request each segment. This enables efficient pre-fetch control to cache servers.
- Since SMIL is based on XML (eXtensible Markup Language) [16], XML-related technologies can be applied to this system.
- 3GPP PSS adopts SMIL as the scene description language. Our dynamic SMIL framework follows the 3GPP-compliant specifications. This is important for deploying this CDN system over 3G mobile networks.
- Content providers can easily create SMIL files for describing streaming content because SMIL is a standard and popular scene description language.

Figure 1 shows the concept of the dynamic SMIL framework. In this system, content providers register their streaming content with SMIL files describing the layout and media synchronization about the content, and do not need to worry about the existence of cache servers. Mobile clients, on the other hand, download the modified SMIL files from a portal server in the CDN. By simply following the downloaded SMIL files, the clients can receive multimedia content from the best cache servers. Even when mobile clients move to another location, the clients can continue multimedia streaming from the best cache servers by updating the SMIL files. The key elements of this system are as follows.

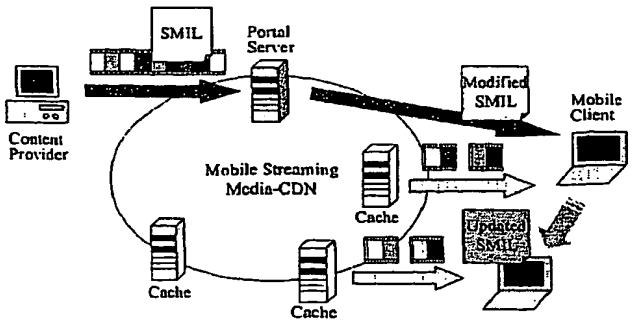


Fig. 1 Concept of dynamic SMIL framework.

1. The CDN provides content segmentation functionality to enable segment caching and to improve cache efficiency without complicating management of the segments. It accepts content registered from content providers and divides the content into several segments if it is too large. At the same time, it modifies the SMIL file corresponding to the segmented content to describe the timing relations among the segments.
2. On-demand URL rewriting described in [17] is applied to SMIL. That is, when clients request a SMIL file, the original content locations in the SMIL file are replaced by the cache server locations, and the modified SMIL file is sent to the clients. The clients, then, request session setup to the cache servers according to the modified SMIL file.
3. To minimize the waste of the network resources and cache memory, streaming segments are scheduled to be pre-fetched just before clients request the segments while the clients are receiving streaming segments. The request times are derived from the timing information described in SMIL files.
4. When mobile clients move from one location to another location, the clients request an update of the SMIL file. The portal server, then, updates the SMIL file to indicate the best cache server locations and creates the SOAP (Simple Object Access Protocol) [18] message that includes the update information from the previous SMIL file. After the clients reproduce the updated SMIL file from the SOAP message, they continue the streaming session according to the updated SMIL file.

The first three items enumerated above are effective in both wired and mobile networks though the last one is related only to mobile networks. Our proposed framework covers all of these features in an integrated

way by SMIL modification.

The rest of this paper is organized as follows. Section 2 explains our architecture overview. Then, each key element to support our CDN architecture, i.e., content segmentation, request routing, pre-fetch scheduling, and session handoff, is described respectively in Sect. 3 through Sect. 6. We report the current status of our prototyping work in Sect. 7, and finally, Sect. 8 concludes this paper.

2. Basic Architecture

The MSM-CDN architecture we propose is shown in Fig. 2. This architecture consists of the wired core network and the radio access network. The CDN overlays the core network, and the CDN control plane is in the middle of the CDN.

The CDN is composed of portal servers, content servers, surrogates, and a content location manager (CLM). The portal servers accept content registrations from content providers and provide content segmentation and SMIL modification functionality. The content servers are simple HTTP (Hypertext Transfer Protocol) [19] and RTSP (Real-Time Streaming Protocol) [20] servers and store a large number of streaming segments derived from the original media content. They do not have caching functionality, and work as HTTP servers for the surrogates and RTSP servers for mobile clients. The surrogates are the RTSP servers that temporarily store streaming segments and serve mobile clients as streaming servers via RTSP session control. The CLM is a server located in the CDN control plane, and it takes charge of managing content locations and scheduling pre-fetches.

The mobile clients can connect to the CDN via the radio access network. They are the RTSP clients that request multimedia content delivery according to the SMIL files sent from the portal server.

From the next section, we describe each key element of our architecture in more detail.

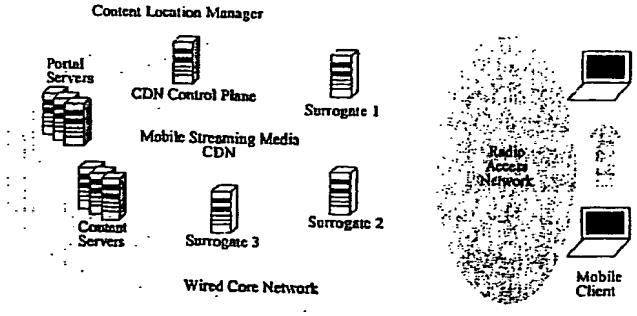


Fig. 2 Basic architecture of mobile streaming media CDN.

3. Content Segmentation

Content segmentation is effective especially for large streaming content. By dividing content into several segments, surrogates can store and remove the content at the granularity of the segments, which leads to efficient cache memory and network resource utilization. Prefix caching [9], selective caching [11], variable-sized segmentation [12], and content-aware segmentation [13] have been proposed as the intelligent variants of segment caching.

We also introduce content segmentation in our CDN architecture to improve cache efficiency. The problem with content segmentation, however, is to complicate the management of these segments. In other words, content segmentation increases the number of the files to be managed. And to realize such intelli[illegible] caching schemes proposed in [9]–[13], the servers in the CDN have to know the relations among the segments. Thus, how to manage the segments is an important issue.

For this purpose, we bind a SMIL file to the segments divided from a streaming content. Since SMIL provides the timing information for playing or displaying, it can easily manage the timing relations among the segments. By parsing the SMIL file, mobile clients and other servers can determine which segments are the next to the segments currently served and how long the segments will last.

The procedure of content segmentation is shown in Fig. 3. First, content providers register their own streaming content with the portal server. Here, the content providers also attach the SMIL files describing the layout and media synchronization about their streaming content. By registering content with SMIL files, content providers can specify how the conte[illegible]s displayed at mobile clients. If the size of the registered content is larger than a certain threshold, the portal server divides the content into smaller segments. At the same time, the portal server modifies the SMIL file attached to the content to describe the timing relations among these segments. Then, it sends the segments to a content server and requires it to store them.

Then, let us see an example of SMIL modification. Here, there is a streaming content whose length is sixty minutes, and its SMIL file is shown in Fig. 4. The SMIL

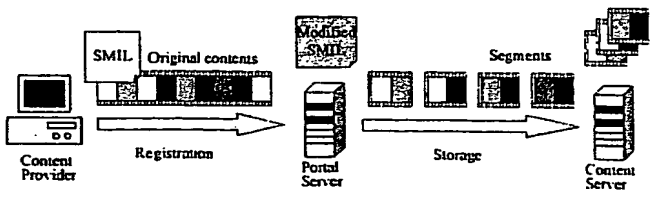


Fig. 3 Content segmentation and SMIL modification at portal server.

shows that content names are "content-A.mp4" for audio content and "content-V.mp4" for video content, and their durations are sixty minutes.

We assume the content is divided into three segments, and each segment is ten, twenty and thirty minutes long, respectively. In this case, the SMIL file is modified at the portal server as shown in Fig. 5. According to the modified SMIL file, these segments are renamed as "content-A-{1,2,3}.mp4" for audio segments and "content-V-{1,2,3}.mp4" for video segments, and the durations of the segments are ten, twenty, and thirty minutes long from the top segments. Since the part located between <seq> and </seq> indicates that these segments are sequentially played and displayed from the top segments, the timing relation among them is evident.

In this way, SMIL files are modified along with content segmentation, and the modified SMIL files indicate the names, durations, and relations of the segments.

```
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN"
"http://www.w3.org/2001/SMIL20/SMIL20.dtd">
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
 <head>
  <layout>
   <region id="v" top="5" left="5" width="180" height="180"/>
  </layout>
 </head>
 <body>
  <par dur="60min">
   <audio src="rtsp://content-server/content-A.mp4" />
   <video src="rtsp://content-server/content-V.mp4" region="v" />
  </par>
 </body>
</smil>
```

Fig. 4 Example of original SMIL file.

```
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN"
"http://www.w3.org/2001/SMIL20/SMIL20.dtd">
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
 <head>
  <layout>
   <region id="v" top="5" left="5" width="180" height="180"/>
  </layout>
 </head>
 <body>
  <seq>
   <par dur="10min">
    <audio src="rtsp://content-server/content-A-1.mp4" />
    <video src="rtsp://content-server/content-V-1.mp4" region="v" />
   </par>
   <par dur="20min">
    <audio src="rtsp://content-server/content-A-2.mp4" />
    <video src="rtsp://content-server/content-V-2.mp4" region="v" />
   </par>
   <par dur="30min">
    <audio src="rtsp://content-server/content-A-3.mp4" />
    <video src="rtsp://content-server/content-V-3.mp4" region="v" />
   </par>
  </seq>
 </body>
</smil>
```

Fig. 5 Modified SMIL file after content segmentation.

4. Request Routing

Request routing is the most important technology for CDN. Several request routing methods have already been proposed [17]. One popular method is DNS-based request routing. However, its resolution is domain level and, therefore, fine-grain request routing at object level is difficult. Another method is using RTSP REDIRECT. Although it's simple, the REDIRECT message has to be issued from original content servers to mobile clients each time the clients request a streaming segment, and this leads to session setup latency.

In our CDN architecture, we apply on-demand URL rewriting [17] to SMIL. Request routing is done by modifying SMIL files at the portal server when mobile clients request the SMIL files. In other words, the portal server replaces the content locations in SMIL files by the surrogate locations, and returns the modified SMIL files to mobile clients. The advantages of this request routing are:

- Request routing is done before clients request a streaming session by RTSP. This enables faster prefetching described in the next section.
- There is no extra procedure such as DNS-based request routing or RTSP REDIRECT method to set up a streaming session between clients and surrogates. This leads to low latency of session setup.

Figure 6 shows the request routing procedure by SMIL modification.

When a mobile client requests a SMIL file to start multimedia content streaming (1), the portal server reads the SMIL file corresponding to the streaming content stored in its memory. Then, the portal server queries CLM about the locations of the segments described in the SMIL file (2). At this time, the portal server notifies the estimated time when each segment will be requested, by parsing the timing information in the SMIL file. CLM utilizes this information for prefetching control described later. The client location (IP address) is also notified to CLM.

CLM selects the best surrogates for each segment

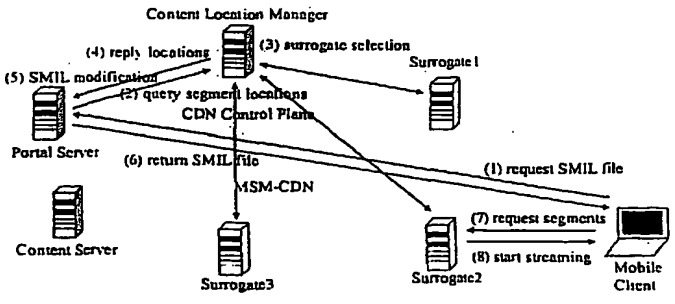


Fig. 6 Request routing procedure by SMIL modification.

```
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN"
"http://www.w3.org/2001/SMIL20/SMIL20.dtd">
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
 <head>
  <layout>
   <region id="v" top="5" left="5" width="180" height="180"/>
  </layout>
 </head>
 <body>
  <seq>
   <par dur="10min">
    <audio src="rtsp://surrogate2/content-server/content-A-1.mp4" />
    <video src="rtsp://surrogate2/content-server/content-V-1.mp4 region="v" />
   </par>
   <par dur="20min">
    <audio src="rtsp://surrogate2/content-server/content-A-2.mp4" />
    <video src="rtsp://surrogate2/content-server/content-V-2.mp4" region="v" />
   </par>
   <par dur="30min">
    <audio src="rtsp://surrogate3/content-server/content-A-3.mp4" />
    <video src="rtsp://surrogate3/content-server/content-V-3.mp4" region="v" />
   </par>
  </seq>
 </body>
</smil>
```

Fig. 7 Modified SMIL file to indicate surrogate locations.

based on the client location, network condition, caching status, surrogate load, and so on (3). In this paper, we do not specify how to select the surrogate locations. After the selection, it responds the locations of the selected surrogates to the portal server (4).

The portal server, then, replaces the original content locations in the SMIL file by the surrogate locations (5). Figure 7 shows an example of the SMIL file whose content locations are modified to indicate the surrogate locations. The original URLs of the segments shown in Fig. 5 are replaced by the surrogate URLs. The locations of the segments could be different. In the SMIL file shown in Fig. 7, surrogate2 is selected for the top four segments, while surrogate3 is selected for the last two segments.

After the modification of the SMIL file, the portal server returns the modified SMIL file to the mobile client (6). The mobile client receives the SMIL file and requests streaming sessions by RTSP according to the SMIL file (7). The surrogate requested to start the streaming session delivers RTP (Real-time Transport Protocol) [21] media packets that include audio and video payloads to the mobile client (8).

As long as the mobile client follows the SMIL file downloaded from the portal server, it always requests session setup to the surrogates selected by CLM.

5. Pre-Fetch Scheduling

To smoothly start streaming sessions, the surrogates have to pre-fetch the segments before the clients request the segments. As described in Sect. 3, the modified SMIL files provide the timing relations among the segments. By utilizing this information, the time when the clients will request each segment can be estimated. Based on the estimated times, pre-fetching is scheduled to be finished before the clients request the segments.

Figure 8 shows the sequence of segment pre-

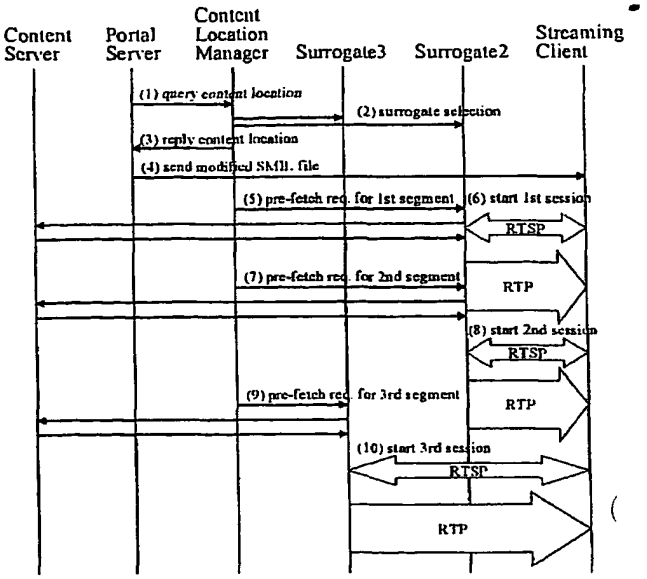


Fig. 8 Sequence of pre-fetching scheduled by content location manager.

Session ID			123456789		
Client IP address			10.20.30.40		
Start time			12:34		
Pre-fetch status			ACTIVE		
#	Time	Dur.	Seg. Name	Orig. Loc.	Surrogate
1	0 m	10 m	content-A-1.mp4	content-server	surrogate2
2	0 m	10 m	content-V-1.mp4	content-server	surrogate2
3	10 m	20 m	content-A-2.mp4	content-server	surrogate2
4	10 m	20 m	content-V-2.mp4	content-server	surrogate2
5	30 m	30 m	content-A-3.mp4	content-server	surrogate3
6	30 m	30 m	content-V-3.mp4	content-server	surrogate3

Fig. 9 Pre-fetch scheduling table.

fetching. In our CDN architecture, CLM takes charge of pre-fetch scheduling.

When the portal server queries CLM about segment locations, it also notifies the timing information for the segments derived from the SMIL file (1). After the surrogate selection is done based on the client location, network condition, caching status, and surrogate load (2), it replies the locations of the selected surrogates (3).

At the same time, CLM creates pre-fetch scheduling table as shown in Fig. 9. The table is composed of session ID, client IP address, start time, pre-fetch status, and pre-fetch information for each segment. Pre-fetch information includes estimated request times, segment durations, segment names, original locations, and surrogates selected to serve the segment.

Based on this table, CLM starts pre-fetching. As shown in Fig. 8, the first audio and video segments are pre-fetched (5) while the client is requesting the segments (6). Before the client finishes receiving the whole

first segments, the second segments are pre-fetched (7). The third segments are pre-fetched to surrogate3 while the client is receiving the second segments from surrogate2 (9).

If the client pauses the streaming session, the surrogate serving the client notifies CLM of the pause. CLM, then, changes the pre-fetch status in the pre-fetch scheduling table to PAUSE and stops pre-fetch requests to surrogates. When the client restarts the streaming, the surrogate notifies CLM of the restart. CLM recalculates the start time and the request times for each segment, and changes the pre-fetch status to ACTIVE. The same procedure is taken when the client skips or goes back to the segments.

6. Streaming Session Handoff

When a mobile client moves from one location to another location, the best surrogate serving it could be changed. One way to redirect the client's request to the new surrogate is sending RTSP REDIRECT from the surrogate currently serving the client. In this case, however, the client will request the next segments from the obsolete surrogate if it is still guided by the SMIL file downloaded when starting the multimedia content delivery. Another way is client-driven session handoff. That is, the client disconnects the current surrogate and requests the new surrogate for a session setup. The problem with this is how the client gets to know the best surrogate locations for itself.

To enable session handoff for all of the streaming segments, the mechanism to update SMIL files is necessary when the best surrogates are changed due to client mobility. In our CDN architecture, a SOAP message is used for this purpose. When a mobile client moves to another location and the best surrogates are changed, the portal server sends the SOAP message that includes the locations of the new surrogates serving the client. The client, then, updates its SMIL file and continues to receive the streaming segments from the new surrogates by following the updated SMIL file.

The advantages of sending SOAP messages to update SMIL files are:

- The SOAP messages can explicitly notify the update of the current SMIL file.
- If the SMIL file to be updated is elaborate for describing the layout and if its size is very large, sending the updated SMIL file itself consumes network resources. In that case, the SOAP message that tells only the update information of the surrogate locations is more cost-effective.
- Both SMIL and SOAP are based on XML. The same modules to parse XML documents can be used at mobile clients. In addition, most of the mobile clients are expected to be SOAP-capable in the near future. This requires only small modification on the clients.

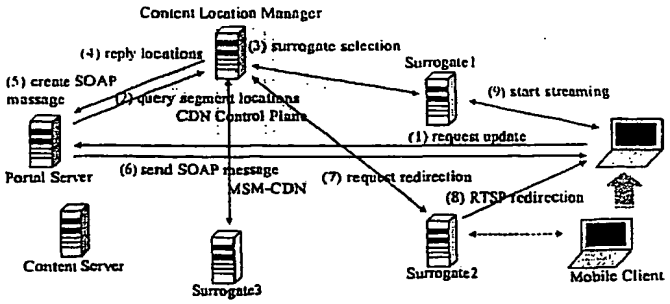


Fig. 10 SMIL update procedure when client moves.

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle=
  "http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <m:update xmlns:m="http://docomo.ne.jp/dynamicSMIL">
   <oldSrc>/smil/body/seq/par[2]/audio/@src</oldSrc>
   <newSrc>rtsp://surrogate1/content-server/content-A-2.mp4</newSrc>
   <oldSrc>/smil/body/seq/par[2]/video/@src</oldSrc>
   <newSrc>rtsp://surrogate1/content-server/content-V-2.mp4</newSrc>
   <oldSrc>/smil/body/seq/par[3]/audio/@src</oldSrc>
   <newSrc>rtsp://surrogate1/content-server/content-A-3.mp4</newSrc>
   <oldSrc>/smil/body/seq/par[3]/video/@src</oldSrc>
   <newSrc>rtsp://surrogate1/content-server/content-V-3.mp4</newSrc>
  </m:update>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig. 11 Example of SOAP message to update SMIL file.

Figure 10 shows the procedure of SMIL update when a mobile client moves. In this figure, we assume the best surrogates are changed from surrogate2 to surrogate1.

When a mobile client moves from one location to another location and detects its movement, the mobile client requests the portal server to update the SMIL file by the SOAP request message (1). After receiving the update request, the portal server requests CLM to recalculate the locations of the best surrogates (2). As similar to the case described in Sect. 4, CLM selects the best surrogates for each segment based on the client location, network condition, caching status, and surrogate load (3). After the selection, CLM responds the locations of the selected surrogates to the portal server (4).

The portal server, then, creates the SOAP response message to inform the client of the new surrogate locations for each segment (5). An example of the SOAP response is shown in Fig. 11. The parts located between <oldSrc> and </oldSrc> indicate the locations to be replaced in XPath (XML Path Language) [22] description. The next parts located between <newSrc> and </newSrc> indicate the URLs of the new surrogates selected by CLM. In Fig. 11, the surrogates serving the second and the third audio/video

segments are changed to surrogate1.

The SOAP message created in this way is transmitted to the client (6). From this SOAP message and the current SMIL file, the client creates the new SMIL file that includes the new surrogate locations. The client continues to request the subsequent segments from the surrogates described in the updated SMIL file.

While CLM responds the segment locations to the portal server (4), CLM also recalculates pre-fetch schedule and rewrites pre-fetch scheduling table. In addition, it may request the surrogate currently serving the client (surrogate2) to redirect the client's request to the newly selected surrogate (surrogate1) (7). Then, surrogate2 issues RTSP REDIRECT method to the client (8), and the client requests surrogate1 to send the segments currently being transmitted from surrogate2 (9). Note that surrogate2 does not need to issue RTSP REDIRECT methods for the subsequent segments because the client requests surrogate1 to deliver them directly according to the updated SMIL file.

This SMIL update mechanism may generate vast signaling traffic if a mobile client moves fast and frequently changes its location. Obviously there is trade-off relation between accuracy of best surrogate locations and the cost of the signaling traffic. In this mechanism, the signaling cost can be flexibly controlled by a counter or timeout mechanism to prohibit the SMIL update for a while. And in an environment where multiple types of radio access networks (RANs) are connected, the change of the RAN type is a good indication for the SMIL update. That is, the best surrogate is expected to be changed with high probability when a mobile client changes its access network, for example, from 3G RAN to a wireless LAN. On the other hand, the best surrogate is expected to be the same as before when a mobile client merely changes its access point in wireless LANs. Some mechanisms to trigger the SMIL update from network to mobile clients may also be possible and more effective if the network knows the coverage area of each surrogate.

7. System Prototype

We have built the first version of MSM-CDN over our mobile QoS testbed called "MOBIQ [23]" as shown in Fig. 12. MOBIQ consists of Diffserv [24]-based IP core network and 3G RAN and IEEE802.11 wireless LAN. The core network is composed of FreeBSD routers with ALTQ [25] module, which provides functionality needed by Diffserv. The 3G RAN includes W-CDMA (Wideband Code Division Multiple Access) RAN emulator [26] and header compression nodes [27] to enable efficient IP packet delivery over 3G wireless links. We have also developed streaming servers, live steaming servers, and mobile clients. All of these streaming servers and clients are compliant with 3GPP-PSS standard.

The MSM-CDN overlay has been prototyped on this testbed. It includes the portal server, the content location manager(CLM), and the surrogates. At this moment, only the request routing procedure shown in Fig. 6 has been implemented. The CLM always selects the closest surrogate to mobile clients by requesting the surrogates to measure round trip times (RTTs) between mobile clients and themselves.

One of the features of this prototype is that all of the interfaces between the above CDN nodes are implemented by SOAP interfaces. Since SOAP is based on XML, these interfaces can be easily extended to include various security features including confidentiality, integrity, and access control. Another advantage of using SOAP interfaces is that content providers or other enterprises connecting to the Internet can easily utilize these interfaces and the CDN functions because SOAP is widely used in the Internet.

We have also developed live streaming splitters in our CDN prototype. In the live streaming case, there is no content to be cached on the surrogates. Instead of the surrogates, the splitters receive live streams from the live streaming servers and copy the streams to their clients. The same request routing procedure can be

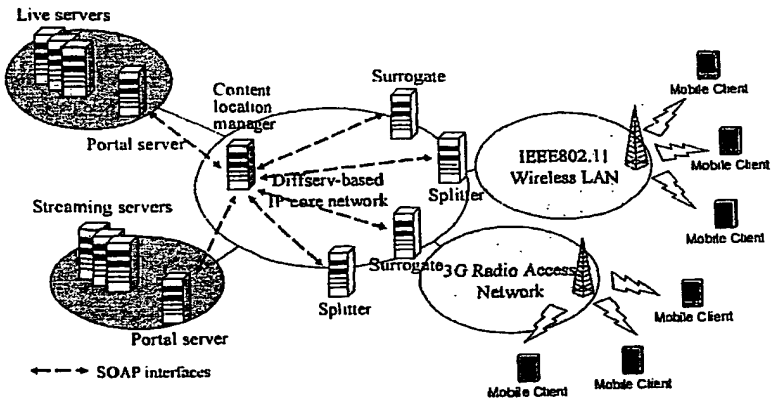


Fig. 12 Prototype of MSM-CDN on mobile QoS testbed.

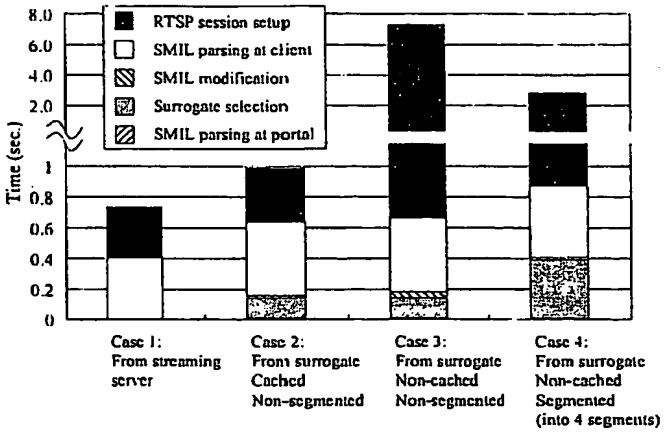


Fig. 13 Time needed for SMIL modification operation and RTSP session setup.

applied to the live streaming.

On this prototype, we can see better media quality with MSM-CDN than without it, when the bottle neck link in the core network is congested. However, mobile clients have to wait more for playback because of the SMIL modification operation including SMIL parsing/modification at the portal server and surrogate selection by CLM. Then, we measured the time needed for the SMIL modification operation as well as the time needed for SMIL parsing at clients and RTSP session setup in several test cases.

Figure 13 shows the results of the above measurement. The test case 1 is a usual streaming case without CDN. In this case, approximately 700 ms was necessary for parsing SMIL at clients and RTSP session setup. The case 2 is a streaming case from the closest surrogate that has already cached the requested content. Compared with the case 1, it took a little bit longer time because the SMIL modification operation was involved. The primary factor of this was the time for surrogate selection. However, we think the time is negligible because it was less than 200 ms. The case 3 is that the requested content whose size is 5.4 MB had not been cached. Here, the bandwidth of the bottle neck link in the core network was 10 Mbit/s. In this case, the time needed for RTSP session setup was nearly seven seconds because the surrogate could not respond RTSP DESCRIBE message until it finished downloading the requested content. On the other hand, the test case 4 where the content was divided into four segments shows that the time for session setup decreased by tolerable level due to shorter download time of the first segment. And our mobile clients could receive and play back the subsequent segments seamlessly. However, we can see slight increase in the time for surrogate selection. This is because the modified SMIL file became complicated and had more URLs in it by content segmentation. Therefore, segmentation of streaming content into unnecessarily many segments may cause bad effect on the waiting time for playback.

8. Conclusions

We presented a mobile streaming media (MSM)-CDN architecture in which all of the technologies related to CDN are enabled by SMIL modification. In this architecture, mobile clients simply follow the SMIL file downloaded from the portal server, and this leads to multimedia content delivery from the best surrogates in the CDN. The key components of this architecture are as follows.

1. To improve cache efficiency, content segmentation is provided along with SMIL file modification to describe the timing relations among the segments.
2. To redirect client requests to the best surrogates, the content locations in the SMIL file are modified to indicate the best surrogate locations.
3. Segment pre-fetching is scheduled just before clients request segments in order to minimize the waste of the network resources and cache memory. The request times are derived from SMIL files.
4. SOAP messages to update SMIL files are defined. This is necessary when mobile clients move and the best surrogates for them are changed.

The current status of our prototype was also reported. We have already built the MSM-CDN over our mobile QoS testbed called "MOBIQ." The MSM-CDN is completely compliant with 3GPP-PSS standard, and all of the CDN nodes have SOAP interfaces to control each other. We also showed that the time needed for the SMIL modification operation was negligible.

We are now investigating segmentation and resource management algorithms. These algorithms require choosing a media segment size for the content segmentation operation. An appropriately chosen segment size will consider client mobility, since each segment boundary provides an opportunity for a session handoff. Segment size also influences content use statistics, since knowledge that users only tend to view the first 30 seconds of a media clip can help the system efficiently allocate its storage resources. Resource management and caching algorithms can also benefit from knowledge of content popularity, since content that is likely to be accessed by other users should be left in cache storage whenever possible.

Interaction with the underlying QoS network is another research topic. Notice that the pre-fetch scheduling table gives a deadline for when the segment must be pre-fetched to the surrogate, however it does not specify a precise time for the pre-fetch to occur. This flexibility allows the CLM and surrogates to schedule the actual pre-fetch time while taking the underlying QoS base network resource usage into account.

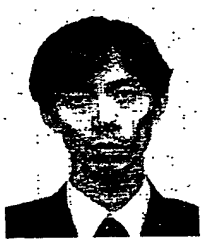
We are also interested in personalized mobile

streaming. Since users have limited devices and time especially in mobile environments, it is important to deliver multimedia content dynamically and automatically summarized depending on user preferences and/or network conditions. Towards this, personalization of SMIL files [28] and MPEG-7 based personalization [29] have already been proposed. We are considering the integration of these personalization technologies into our MSM-CDN architecture.

References

[1] Akamai Technologies, Inc., http://www.akamai.com/
[2] NTT DoCoMo, Inc., i-mode, http://www.nttdocomo.co.jp/english/i/index.html
[3] WAP Forum, Wireless Access Protocol, http://www.wapforum.org
[4] NTT DoCoMo, Inc., FOMA (Freedom Of Mobile multimedia Access), http://foma.nttdocomo.co.jp/english/
[5] 3GPP TS 26.234 v 4.1.0, Transparent end-to-end packet switched streaming service (PSS); Protocol and codecs, Sept. 2001.
[6] R. Rejaie and J. Kangasharju, "Mocha: A quality adaptive multimedia proxy cache for Internet streaming," Proc. NOSSDAV 2001, 2001.
[7] R. Rejaie, H. Yu, M. Handley, and D. Estrin, "Multimedia proxy caching mechanism for quality adaptive streaming applications in the Internet," Proc. INFOCOM 2000, 2000.
[8] J. Kangasharju, F. Hartanto, M. Reisslein, and K.W. Ross, "Distributing layered encoded video through caches," Proc. INFOCOM 2001, 2001.
[9] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," Proc. INFOCOM'99, 1999.
[10] E. Bommaiah, K. Guo, M. Hofmann, and S. Paul, "Design and implementation of a caching system for streaming media over the Internet," Proc. Real-time Technology and Applications Symposium (RTAS 2000), 2000.
[11] Z. Miao and A. Ortega, "Proxy caching for efficient video services over the Internet," Proc. Packet Video Workshop 1999, 1999.
[12] K.-L. Wu, P.S. Yu, and J.L. Wolf, "Segment-based proxy caching of multimedia streams," Proc. 10th International World Wide Web Conference, May 2000.
[13] S.-J. Lee, W.-Y. Ma, and B. Shen, "An interactive video delivery and caching system using video summarization," Proc. WCW 2001, 2001.
[14] C.D. Cranor, M. Green, C. Kalmanek, D. Shur, S. Sibal, J.E. Van der Merwe, and C.J. Sreenan, "Enhanced streaming services in a content distribution network," IEEE Internet Computing, vol.5, no.4, pp.66–75, July Aug. 2001.
[15] Synchronized Multimedia Integration Language (SMIL 2.0), W3C Recommendation, Aug. 2001. http://www.w3.org/TR/smil20/
[16] Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation, Oct. 2000. http://www.w3.org/TR/REC-xml
[17] A. Barbir, B. Cain, F. Douglis, M. Green, M. Hofmann, R. Nair, D. Potter, and O. Spatscheck, "Known CDN request-routing mechanisms," IETF Internet-Draft, draft-cain-cdnp-known-request-routing-03.txt (work in progress), Nov. 2001.
[18] Simple Object Access Protocol (SOAP) 1.1, W3C Note, May 2000. http://www.w3.org/TR/SOAP/
[19] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol—HTTP/1.1," IETF RFC 2616, June 1999.
[20] H. Schulzrinne, A. Rao, and R. Lanphier, "Real time streaming protocol," IETF RFC 2326, April 1998.
[21] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," IETF RFC 1889, Jan. 1996.
[22] XML Path Language (XPath) Version 1.0, W3C Recommendation, Nov. 1999. http://www.w3.org/TR/xpath
[23] T. Yoshimura, T. Ohya, H. Matsuoka, and M. Etoh, "Design and implementation of mobile QoS testbed MOBIQ for multimedia delivery services," Proc. Packet Video Workshop 2002, April 2002.
[24] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," IETF RFC 2475, Dec. 1998.
[25] K. Cho, "Managing traffic with ALTQ," Proc. USENIX 1999, June 1999.
[26] H. Inamura, T. Ishikawa, and O. Takahashi, "Evaluation of TCP traffic over W-CDMA network," IPSJ Technical Report, MBL18-33, Sept. 2001.
[27] T. Yoshimura, T. Kawahara, T. Ohya, and M. Etoh, "Multiple-reference compression of RTP/UDP/IP headers for mobile multimedia communications," IEICE Trans. Fundamentals, vol.E85-A, no.7, pp.1491–1500, July 2002.
[28] R. Hjelsvold, S. Vdaygiri, and Y. Leaute, "Web-based personalization and management of interactive video," The 10th International World Wide Web Conference, May 2001.
[29] S. Sekiguchi, M. Etoh, K. Emura, W. Fujikawa, K. Masumitsu, and T. Echigo, "Video digest delivery using MPEG-7 media structure description and its authoring system," IEICE Technical Report, PRMU2001-92, Sept. 2001.

Takeshi Yoshimura received the B.E. and M.E. degrees from Department of Information and Communication Engineering at the University of Tokyo in 1997 and 1999, respectively. He joined NTT Mobile Communication Network, Inc. in 1999. He is currently a research engineer of Multimedia Laboratories, NTT DoCoMo, Inc. His current research interests include mobile streaming media technology and content delivery network architecture. He is a member of IEEE and IPSJ.



Yoshifumi Yonemoto received the B.E. and M.E. degrees in electrical and computer engineering from Nagoya Institute of Technology, Nagoya, Japan, in 1990 and 1992, respectively. In 1992 he joined the Matsushita Electric Industrial Co., Ltd., and in 1999 he transferred to Matsushita Communication Industrial Co., Ltd. Since 2001, he is with NTT DoCoMo Multimedia Laboratories. His current research includes mobile multimedia content description and delivery architecture.

Tomoyuki Ohya received the B.E. and M.E. degrees in electronic engineering from Kyoto University in 1986 and 1988 respectively, and the M.S. degree in management of technology from Massachusetts Institute of Technology in 2000. He joined NTT in 1989, and since 1992 he has been working at NTT DoCoMo, Inc. He has been engaged in research and development of digital speech coding technologies for PDC and IMT-2000. Since 1996, he has been participated in the MPEG-4, ETSI and 3GPP international standardization activities, and mainly focusing on the research and development of digital signal processing technologies for the 3rd generation mobile communications systems. Currently his main research interest is on the multimedia signal processing and QoS architecture for 4th generation mobile communications networks. He is a member of IEEE. He received the Young Engineer Award from IEICE in 1995.

Minoru Etoh received the B.E. and M.S.E.E. degrees from Hiroshima University, Hiroshima, Japan, in 1983 and 1985, respectively. He received the Ph.D. degree from Osaka University, Osaka, Japan, in 1993. In 1985 he joined the Matsushita Electric Industrial. Co., Ltd. From 1988 to 1990 he worked for the ATR Communication Systems Research Laboratories, The Advanced Telecommunication Research Institute International, Kyoto. From 1991 to 1993, he was a Visiting Researcher at Osaka University. From 1994 to 1998, he was at the Central Research Laboratories of Matsushita Electric, and in the meantime he participated in the MPEG-4 standardization. He was also Adjunct Professor of Nara Institute of Science and Technology from 1996 to 2000, and Adjunct Lecturer of Osaka University from 1999 to 2001. In 2000, he moved NTT DoCoMo Multimedia Laboratories. He is now Director of Multimedia Signal Processing Laboratory. His current research includes media coding/processing, distributed concurrent software, pattern recognition, and networking. He received the 1995 Best Paper Award of IEICE Japan, the 14th Telecom System Technology Prize of the Telecommunications Advancement Foundation (1998), the 7th Sakai Commemorative Prize of IPSJ (1998), respectively. He is a member of IEEE and IPSJ.

Susie Wee received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology in 1991, 1991, and 1996, respectively. She performed her master's thesis work at the Jet Propulsion Laboratory in deep-space optical communication; she received a NASA Group Achievement award for this work. At MIT, her doctoral research was in the area of scalable video coding, video communications over broadcast channels, and high-definition television system design. In 1996, she joined HP Labs as a research scientist. Since 1999, she has been the R&D project manager of the HP Labs Streaming Media Systems Group. Her current research interests are in media delivery to diverse clients over packet networks; efficient video coding, transcoding, and compressed-domain processing algorithms; media-aware networking for next-generation wired and wireless networks; and mobile streaming media system design. In addition to working at HP Labs, she has been a consulting assistant professor at Stanford University since 1999 where she co-teaches a graduate-level course on digital video processing. She is currently an associate editor for the IEEE Transactions on Image Processing. She received Technology Review's Top 100 Young Investigators award in 2002.